Border Gateway Protocol

Glue of the Internet

Felix G. Carapaica

First Edition

Border Gateway Protocol, Glue of the Internet

ISBN 978-0-9939150-2-4

9 780993 915024 >

ISBN: 978-0-9939150-2-4

Cover Page: DNA helix of BGP Clouds. Design and illustration by Daniela Carapaica.

Chapters

Chapter 1: Introduction to BGP

Chapter 2: BGP Basic Mechanisms

Chapter 3: BGP Attributes

Chapter 4: BGP Inside the Autonomous System

Chapter 5: Techniques for BGP Policies

Chapter 6: Controlling Traffic between AS

Chapter 7: Community Attribute

Chapter 8: Aggregation of Routes

Chapter 9: Enterprise Models to Access the Internet

Chapter 10: BGP Multi-home Organization

Chapter 11: BGP as a Signalling Protocol

Chapter 12: BGP in the Data Centre

Table of Contents

Chapter 1 Introduction to BGP

Preface 10

History of BGP in the Internet 1

- BGP Message Exchange Principles 6
- BGP Autonomous Systems 6
- BGP Basic Configuration 7
  - BGP Basic Configuration for Cisco IOS 10
  - BGP Basic Configuration for Juniper JUNOS 12
  - Configuration Scenario for both Cisco IOS and Juniper 16
- BGP Basic Conversation 18
- Configuration of BGP to carry IPv6 routing information 22
- An Overview of BGP in the Internet Context 27

Chapter Exercises 34

- Practice Basic BGP 34
  - Practice Basic BGP 37
  - Labs 39
- Chapter References 43

Chapter 2 BGP Basic Mechanisms

BGP Encapsulation and Protocol Header 55

- BGP OPEN Message 57
- BGP UPDATE Message 60
- BGP NOTIFICATION Message 63
- BGP KEEPALIVE Message 67
- BGP ROUTE REFRESH Message 69
- BGP Finite State Machine FSM 71

Chapter Summary....72
BGP Mechanisms....73
Exercise....73
Chapter References....74

Chapter 3 BGP Attributes

Structure of a BGP Attribute....75
Flags Field....76
Optional flag bit....76
Optional flag bit....78
Flag bit transitiveness....78
Flag bit Completion....79
Flag bit Length....79
BGP Attribute type field....80
BGP Attribute ORIGIN....81
BGP Attribute AS_PATH....86
Analysis of AS_PATH Attribute....88
Use of AS_PATH to break routing loops....93
BGP Attribute NEXT_HOP....94
Chapter Summary....96
Exercise....97
Configuration of network for practice and analysis....97
Chapter References....101

Chapter 4 BGP inside the Autonomous System

Fundamentals of iBGP 102

Internal BGP (iBGP) 106

iBGP inside the AS core 112

Attaching an AS number to the AS_PATH rule 120

One-hop iBGP rule 121

iBGP split horizon with poison reverse rule 122

iBGP in Full Mesh Configuration in the Core 122

iBGP Route Reflector 124

Core Free of iBGP 128

Chapter Summary 133

Exercises 134

Exercise Route Reflection 134

Exercise Route Reflection 136

Exercise iBGP and Route Reflection 137

Exercise iBGP and Route Reflection 138

Exercise iBGP and Route Reflection 139

Chapter Reference 140

Chapter 5 Techniques for BGP Policies

BGP Default Decision Process 141

Modifying BGP Default Behaviour with Policies 148

Implementing Policies in BGP: the basic tools 152

Regular Expression in JUNOS 171

Implementing BGP Route Filtering 175

Cisco IOS Distribute-list 176

Cisco IOS Filter-list 179

Cisco IOS Prefix-list 181
Cisco IOS Route-map 186
Juniper JUNOS Policy Configuration 193
JUNOS Route-Filter 194
Routing Policy Specification Language RPSL 201
RPSL structure 202
Describing Policies with RPSL 203
RPSL Examples 203
Chapter Summary 210
Exercise 211
Chapter References 221

Chapter 6 Control of Traffic between AS

Default Behaviour of a Network Topology 222
Using Cisco Weight to modify outbound paths 233
Using Local Preference Attribute to modify paths 234
Cisco Configuration 234
Juniper Configuration 236
Using Multi-Exit-Discriminator to modify paths 238
Cisco implementation 238
Juniper implementation 240
Using AS-PATH to change path appeal 241
Cisco implementation 241
Juniper implementation 243
Chapter Summary 244
Exercises 245
Appendix 249

Chapter References 257

Chapter 7 Community Attribute

Format of the Community Attribute 258
Standard Community 259
Community Attribute Case Topology 259
Implementing Policies with Communities 266
Well-Known Communities 269
Do Not Export 269
Do Not Advertise 271
Manipulating traffic paths with Communities 272
Chapter Summary 277
Exercises 278
Chapter References 282

Chapter 8 Aggregation of Routes

Aggregation of IP Route Addresses 283
A Technique to Allocate Address Space 288
Configuration of IPv4 Route Aggregation 310
Customer Advertises Aggregation plus Details 313
Customer Advertises Aggregation Only 314
Cisco Implementation 314
Juniper Implementation 316
Aggregation at the Provider Level 317
Aggregation by Policy at the Provider Level 319
Aggregation and BGP Attributes 327
AS Path Attribute 327
Atomic Aggregate Attribute 329

Aggregator Attribute 330
Aggregation to control inbound traffic 333
Aggregation of IPv6 Address Space 337
IPv6 Address Scheme 337
IPv6 Address Structure 339
Configuration of BGP to Aggregate IPv6 address space 350
Chapter Summary 362
Exercises 363
Chapter References 370

Chapter 9 Enterprise Access to Internet

Client Enterprise gets simple Internet Access from ISP 371
Two AS peering to exchange local routes 380
Multiple paths between two autonomous systems 383
Cisco IOS Implementation 384
Junos Implementation 388
Chapter Summary 393
Chapter References 393

Chapter 10 BGP Multihome Organization

Network topology 394
Topology general description 395
Configuration 396
AS 5664 Sheridan College 396
Orion AS 26677 406
Router R5, AS 16475, Brock University 410
Configuration Orion and TorIX 413
TorIX AS 11670 414

Router R2 in AS 11670 414
Router R3 in AS 11670 415
Router R6 AS 5664 Continuation 416
AS 11670 TorIX 418
Routers R9 and R10 in AS11670 419
R7 in AS 5664 420
Router R7 in AS 5664 421
Router R8 in AS 5664 423
Cogent AS 174 424
Verification 425
Chapter Summary 438
Exercises 439
Appendix 441
Case Topology and basic skeleton configuration 441
Chapter References 446

Chapter 11 BGP as a Signalling protocol

The functioning of a Service Providing Core 447
Configuration of an MPLS L3VPNv4 core 463
Analysis of the BGP Conversation 471
BGP signalling Layer 2 information 478
BGP signalling Layer 2 Virtual Private Networks 478
Virtual Private LAN Service (VPLS) 479
The role of BGP in VPLS 480
BGP signalling EVPN 487
BGP EVPN Conversation 491
BGP signalling Security Rules with FlowSpec 498

Chapter References 503

Chapter 12 BGP in the Data Centre

BGP in the Leaf and Spine Topology 504

BGP behaviour in the Leaf and Spine Topology 506

Configuration of BGP in the Quagga Router 517

Analysis of transactions between nodes C1 and C3 521

Chapter Summary 532

Chapter References 533

Preface

This textbook was born out of necessity to teach my class of IP Engineering at a College level. Although there is a bounty of documentation about Border Gateway Protocol, it is often found in compartments and it typically focuses on a vendor way of configuring the protocol. More than anything, I wanted to have one self-contained text that would cover most of the fundamentals of the BGP protocol.

Furthermore, the content needed be covered with a balance of theory and practice. Network professionals have to continuously update their knowledge by both understanding the fundamental building blocks behind the functioning of a protocol and also by being able to actually deploy network topologies. In short, this audience needs theory and its application.

In this book, the BGP story is told from its origins, then going through its application in several cases. Practical situations are used to learn not just the configuration ways of major vendors such as Cisco and Juniper, but also to extract the theoretical background by sniffing and dissecting the BGP conversations between routers.

Furthermore, since I believe that learning by doing is the best venue for technology related fields, this text has plenty of exercises and examples explained in detail. Almost all of them can be implemented in network simulation software such as GNS3 and others.

This text was destined from the beginning to be an eBook. It makes sense that a topic about the Internet to be written in an electronic format. So, it was not meant to be printed. That was a liberating decision because it freed me to explain with pictures, diagrams, and colours as much as I could because a picture is worth a thousand words and there more than 500 figures here.

CHAPTER 1

Introduction to BGP

The Internet is a collection of thousands of generic clouds representing IP networks. Internally, each of these clouds might contain thousands of devices, links and hosts. Individually, each cloud is ruled by its own administration. That is what defines an internal routing domain.

Altogether, the Internet is based on the interconnection of thousands of these individual routing domains. Remarkably, there is not a central control to rule over all these clouds. Instead, the functioning of the Internet is achieved by loose consensus, agreements and the following of best practices by the administrations of each routing domain.

Since these domains are intertwined in a web of connections they need a language, a routing protocol, to carry on the conversations that will spread the knowledge of where to find IP network destinations. This language is the Border Gateway Protocol BGP.

History of BGP in the Internet

It is necessary to briefly recall the events that led to the development of BGP to understand how the protocol turned out to be as it is. The current Internet was born out of a small number of academic and research networks. In 1968, [1] the **Defence Advanced Research Projects Agency** (**DARPA**) sponsored a bidding competition to develop a network to interconnect the research centres located in universities in the United States. The contract was won by the company BBN (Bolt, Beranec and Newman) which started the design and building of the first packet switch named the Interface Message Processor IMP. In 1969, a pair of IMPs were deployed to interconnect the computer networks of UCLA and Stanford University giving origin to the ARPAnet [1, 2]. Soon, other organizations would be joining in this embryo of Internet.

This was a connection oriented, reliable, packet switching network. Later, this network model would change dramatically with the introduction of the concept of Open System architecture by Vinton Cerf and Robert E. Kahn. This new paradigm lead to the specification [3] in 1974 of the TCP protocol. By 1984, the ARPAnet was fully converted to the use of TCP for transport control and IP for network data delivery; effectively giving birth to the modern Internet.

Eventually, a major problem became evident. The growing number of sites or network domains being connected to the new Internet could not be sustained without the proper routing protocol. Internally, many networks were running RIP and manually defined (static) connections. Externally, the networks ran Exterior Gateway Protocol EGP, a protocol based on link state advertisements [4] that were used to calculate the most efficient forwarding topological tree. Link state based routing systems are efficient inside routing domains of limited size, but they are not the right choice for a large intermesh of dissimilar networks. By the 1980s, this model was showing scaling problems so it became clear that a new routing protocol would have to be designed. In 1984, Yakov Rekhter, and Kirk Lougheed attended an **Internet Engineering Task Force (IETF)** meeting and they seized this opportunity to brainstorm ideas to find a solution.

It has become almost a legend the story of how they sketched on napkins a basic design of a protocol during an informal gathering. Days later, Yakov Rekhter refined these drawings on a second sketch [5] that has been saved for history (it is on display at Bell Labs, NJ).

"3-napkings protocol"

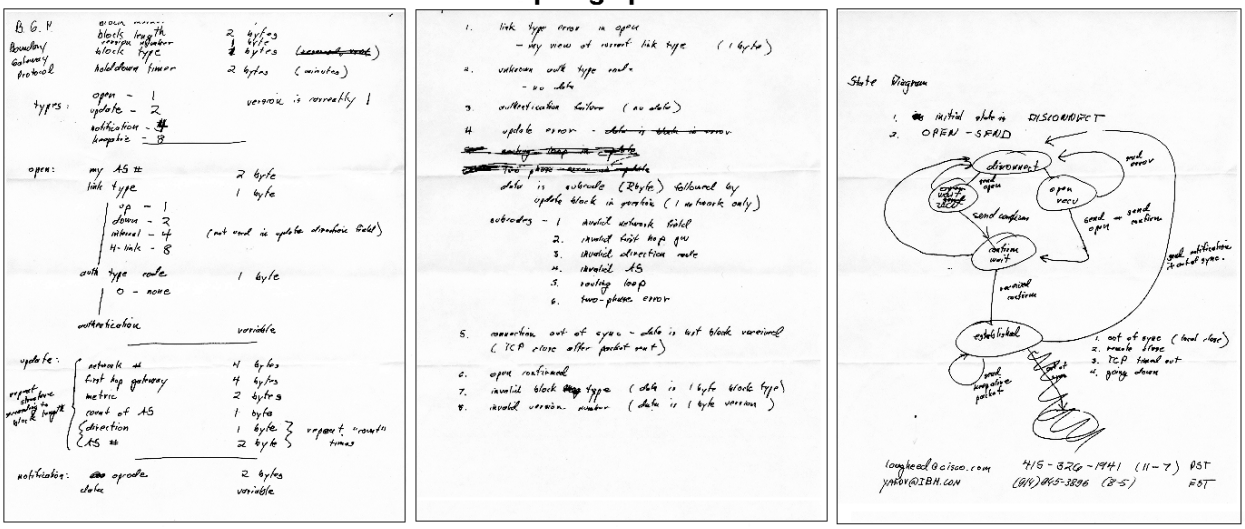


Figure 1.1 the 3-napkin protocol, courtesy of Yakov Rekhter. [5]

It is remarkable that only 3 pages were enough to contain the core of the protocol. These brief pages described the messages needed, size in bytes, functions and the finite state machine. Interestingly, the original name given was not Border but Boundary Gateway Protocol. Eventually, Rekhter became known as the father of BGP as he has authored many of the specification (Request for Comments RFC) documents together with Tony Li [8].

Figure 1.2 Yakov Rekhter [6]

The protocol was born in the spirit of the Internet design motto: "rough consensus, working code" and from the very beginning, the designers made several [8] fundamental design considerations:

- The new protocol had to solve practical, immediate problems.

Due to the pressing need to come up with a solution, it was clear that the new protocol had to concentrate only in the routing process. A long, big design was not a possibility. This consideration had a great influence in the choosing of the encapsulation of the protocol. The next diagram shows all the different design possibilities of a routing protocol.

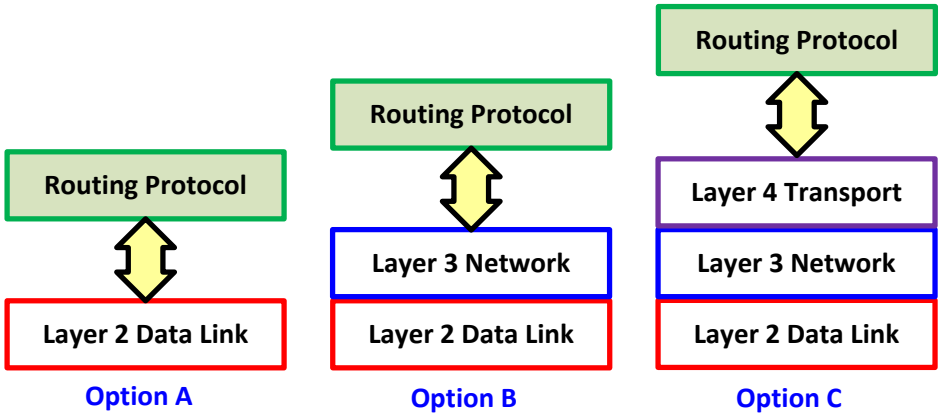


Figure 1.3

Option A places the routing protocol directly on top of layer 2. That implies that the functions that would be provided by layers 3 and 4 still have to go to some other place, obviously into the routing protocol, making it more complex. Furthermore, choosing this direct encapsulation implies that all existing layer 2 protocols would need to be updated to support the new protocol TYPE and that is a complicated process. There is only one routing protocol that uses this encapsulation model and that is ISO's Intermediate System to Intermediate System (IS-IS).

Option B implies that the protocol would be on top of a layer 3 protocol such as IPv4 or IPv6. That means that there is no layer 4 protocol to offer the function of transport control to the topmost protocol. Nonetheless, the need for this transport role still exists and it has to be provided somehow. Hence, parts of the transport function have to be incorporated inside the routing protocol. Tasks such as neighbour discovery, opening a conversation, keeping the pace of it, and closing it. Also data sequence control and error checking. Subsequently, adding those tasks to the routing protocol makes it more complicated to design and build. There are two routing protocols that use this model: OSPF and EIGRP. In the case of OPSF [9] a big part of the protocol engine is dedicated to keeping track of the conversation with the neighbour routers.

The final option C ports the routing protocol on top of a transport (layer 4) protocol. There are two choices here, either unreliable connectionless UDP or reliable connection oriented TCP. Only one routing protocol uses UDP and that is RIP. This routing protocol just requires basic transport without reliability requirements for small size routing domains, so UDP is adequate to do just that. However, in the case of BGP, very large routing tables with hundreds of thousands of prefixes were expected, and that would require a reliable connection oriented transport protocol that would keep track that the routing information was delivered complete, in order and that it was acknowledged back. TCP is the right choice to provide such function.

Consequently, it was decided that TCP would provide reliable connection management services to BGP and this decision has its effects. To establish a session, TCP requires that the IP addresses of the end nodes be known. So BGP would not have automatic neighbour discovery*. Instead, the peer's IP address must be supplied by the network administrator during the configuration process.

Also, by choosing TCP as the transport protocol, routers can establish BGP sessions without the requirement of being directly connected. There is another consequence yet, BGP can send and receive huge amounts of routing information thanks to the flow control provided by TCP. No other routing protocol can handle the amount of routing items than BGP can. Finally, by using something that was already available, the design of BGP was greatly simplified and accelerated. Time has proven that porting BGP on top of TCP was the correct decision by the designers.

* This default behaviour has been modified lately for certain applications. There are now BGP auto-discovery capabilities for special implementations in data centres mostly.

- What would BGP have to advertise?

Solving the problem of connectivity inside the routing domains was left to the Internal Gateway Protocols IGP such as OSPF, IS-IS, RIPv2 and EIGRP. On the external side, the new protocol would abstract the internal topological details of the connected networks. From the viewpoint of BGP, the internal details were not as relevant as to finding paths to the destinations.

This lead to a momentous idea, that the routing domains could be viewed as "abstract clouds" containing routing information to be advertised by BGP. These clouds, or **Autonomous Systems** as they were named, would receive **unique number identifiers**. If the concept is pushed even further, these "clouds" just contain **objects** with **attributes** where the object can be an IPv4 route, an IPv6 route, or it could be something else. Significantly, this design made BGP to be **Multiprotocol** for it can carry information about many things, not just IPv4 routes.

The insight of Rekhter et al of looking at the Internet as a collection of interconnected clouds containing objects with properties and designing BGP in consequence, has resulted in the continuous relevance of BGP over time as new applications have been found for the protocol.

Currently, BGP is being used not just to keep the Internet connected for both protocols IPv4 and IPv6, but BGP is also used by carriers to provide various services to customers with multiple MPLS provisioning models (L3VPN, L2VPN, VPLS), by data centres to advertise information about virtual machines including MAC addresses and VLAN information (eVPN model) by Internet Service Providers to advertise and enforce security rules (Flow Spec) and by software designers to push configuration files (ExaBGP).

BGP Message Exchange Principles

BGP was designed with a lot of common sense. From the viewpoint of a BGP speaking router, there are only four fundamental functions that are realized with four messages.

Function	BGP Message
Begin a session with another peer router.	**OPEN**
Exchange routing information.	**UPDATE**
Notify when things go wrong (errors and mismatches)	**NOTIFICATION**
Keep the session **up** with small periodic messages.	**KEEPALIVE**

These are the four original BGP [8] messages. There was no need for more. Later, a fifth message, **ROUTE REFRESH**, was added to notify changes in inbound policies.

BGP Autonomous Systems

The internet is a collection of network-clouds that group ranges of IP prefixes and these clouds exchange routing knowledge about IP prefixes using BGP. These Internet clouds are called **autonomous systems (AS)** since each one has its own administration and policies. In consequence, these clouds also need to have **identifications** to tell them apart.

The Autonomous System Numbers ASNs numbers are assigned by the Internet Assigned Number Authority IANA to Regional Internet Registries (RIR) and these RIRs assign the AS numbers as per request. The original AS numbers range from 0 to 65,535. This is the result of the binary combination of a 2 bytes long field in the BGP OPEN message. There are ASNs which are reserved for special cases which are the following:

- The first ASN (0) and the last (65,535) cannot be used.
- The numbers from 64,512 to 65,534 are for private use only.

This numbering scheme would limit the Internet to have 65,510 global autonomous systems and the expectation that the original ASN range will be depleted eventually. This lead to the expansion of the range with the introduction of 4 bytes long ASN. That yields more than four billion AS numbers, to be exact 4,294,967,296.

- The numbers from 4.2 billion to 4,294,967,295 are reserved for private use.

BGP Basic Configuration

In this section, the BGP configuration philosophy is explained before getting into specific configuration commands. Independently of the vendor equipment, there are configuration actions that are typical because of the way BGP works. If the protocol functioning is well understood, the network administrators can figure the commands out across vendor platforms. Later, in the next sections, the configuration specific details from two major vendors, Cisco and Juniper, are shown.

The following figure shows two autonomous systems AS # 1 and AS # 2. These numbers have been chosen because they make this explanation simpler, but both AS numbers belong to actual Internet organizations and should not be used in any other situation. The internal details of the AS are not important right now, so it suffices to say that AS1 advertises only one network 192.168.1.0/24 and that the other autonomous system, AS2, advertises the network prefix 192.168.2.0/24.

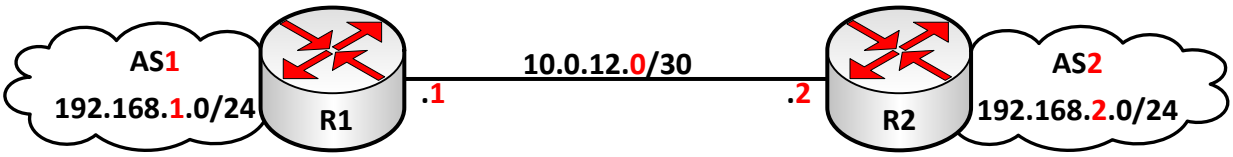


Figure 1.3

The router sitting at the edges of the clouds are aptly named **Autonomous System Border Routers (ASBR)** and they talk BGP to each other. Their role is to exchange the network prefixes and their autonomous system location. An ASBR has to be aware of the Autonomous System of which it is part of. Consequently, the BGP configuration typically begins by defining the local AS number.

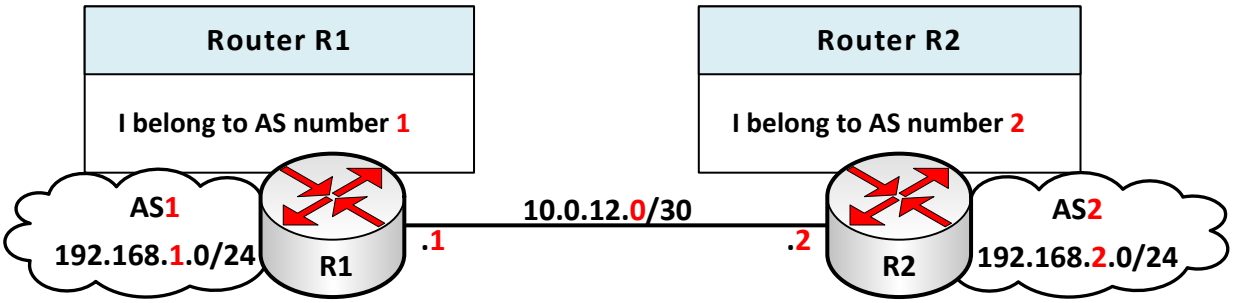


Figure 1.4

In a conventional deployment, BGP does not have a mechanism to discover neighbours, nor should have one (you do not want rogue routers peering with your ASBR). Consequently, the network administrators have to deliberately configure the routers pointing to the destination of the other neighbour (or peer) router. Also, the administrators have to specify the autonomous system number of the peer. Thus, the configuration of BGP is a two way relationship.

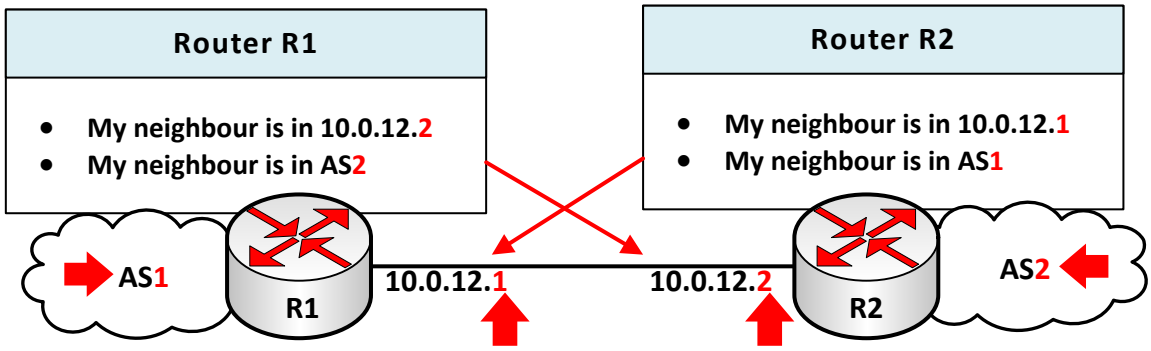


Figure 1.5

A BGP session can be established without any authentication, but for security reasons, the BGP messages should be validated for authenticity. To implement that, both peering sides share a common secret password and a hashing algorithm (MD-5) that creates one signature digest per BGP transaction. Each TCP segment carries one MD-5 hash for the particular BGP message being delivered. It is necessary to clarify that the BGP messages are in clear-text format (not ciphered at all). If a BGP message were intercepted and maliciously modified, the content of the message and the hash would not match when the message is interpreted by the receiving router, so the message would be ignored.

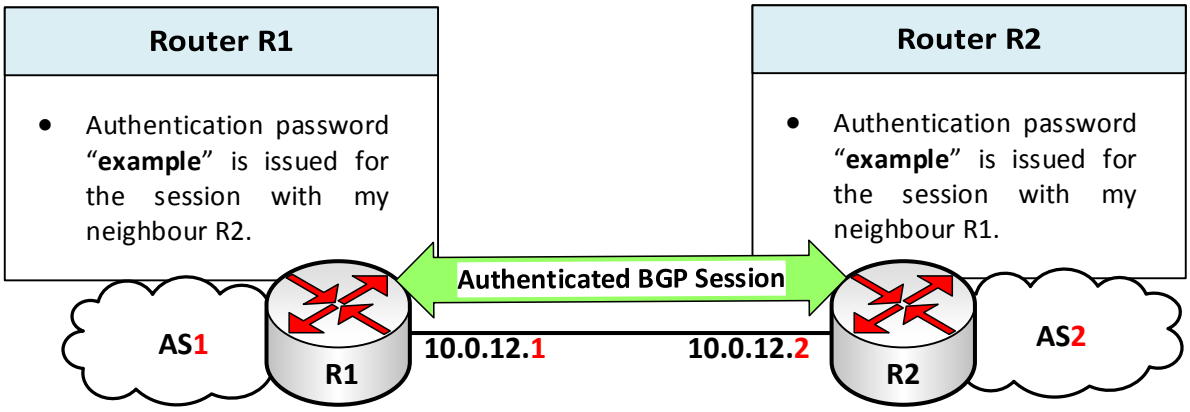


Figure 1.6

The previous configurations steps have to be done independently of the vendor equipment being used. Now, the two routers can peer with each other, but that does not mean that they can exchange routing knowledge just yet. Additional configuration is required because routing knowledge transference is done in two basic ways, by **origination** and by **propagation**. **Origination** is the initial **injection** of the IP prefixes into the global Internet routing table by the autonomous system of its origin. **A BGP router can only originate a route if it is present in its own routing table**. That makes sense since a router should not advertise a black hole. This is implemented in several ways:

a) Use of a manual command to match a specific local IP prefix (network command).
b) Use of redistribution from an IGP such as EIGRP, OSPF, RIP, ISIS.
c) A manual entry or re-origination.

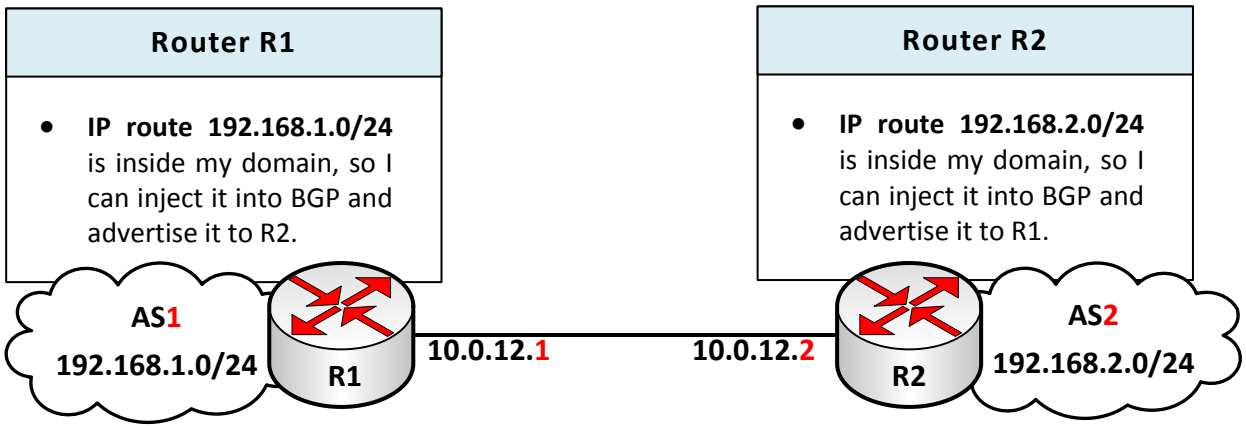


Figure 1.7

In summary, origination requires the conscious decision of the domain (AS) administrator to advertise the prefix and a command is to be issued to match an entry that exists in the local IP routing table; otherwise the prefix will not be advertised. Again, BGP must not advertise what it cannot be reached.

Additionally, BGP **propagates** the routing knowledge that has come from other autonomous systems. That is a dynamical default behaviour of BGP that can be modified by policies and filters, but basically:

a) A BGP router re-advertises what it has learned from other BGP routers. This rule is observed when more routers and AS are added to the previous topology.

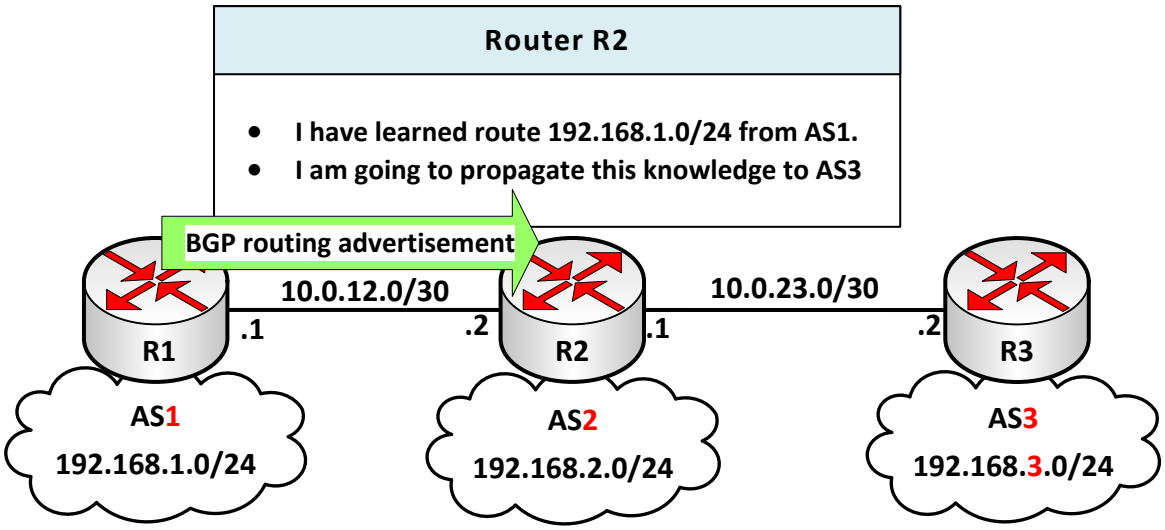


Figure 1.8

b) A BGP router might not re-advertise what it has learned from other BGP routers if there is an administrative action that prevents doing so. This is the main idea behind **routing policies**.

Once the general philosophy of BGP configuration has been outlined, it is time to describe the specific commands used by the vendors Cisco and Juniper.

BGP Basic Configuration for Cisco IOS

This is the Cisco configuration that defines the BGP process in router R1.

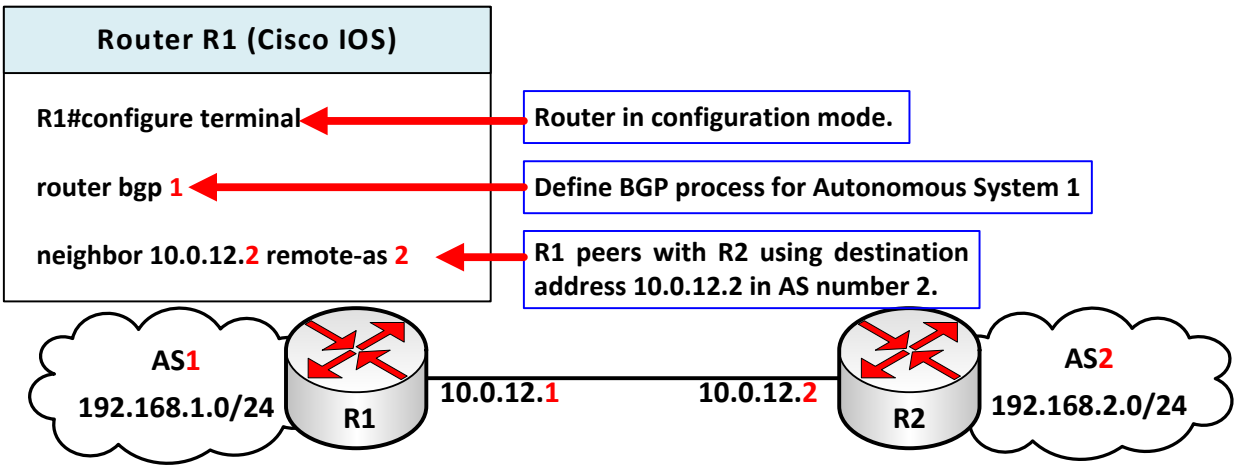


Figure 1.9

Consequently, the initial configuration for the two routers is shown next:

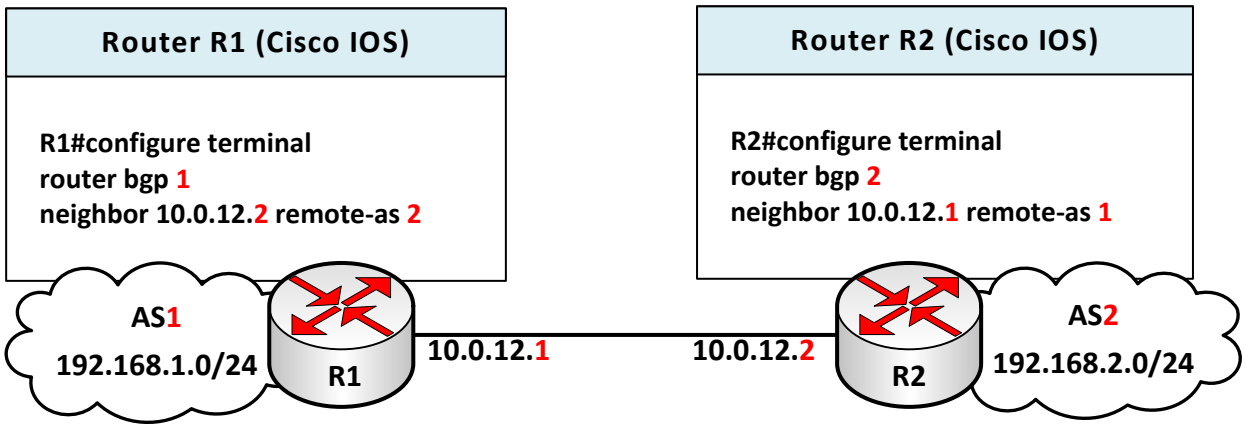


Figure 1.10

Next, the routing knowledge is originated by using the network command. The networks located inside the autonomous system are going to be advertised when the "network" command injects the knowledge into the BGP process.

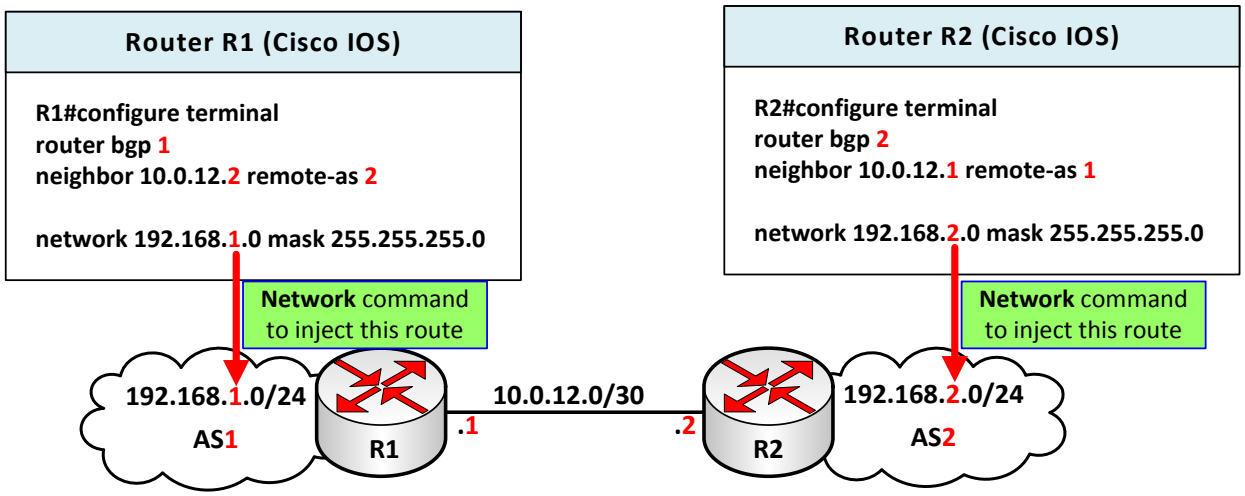


Figure 1.11

BGP Basic Configuration for Juniper JUNOS

Suppose now that R2 (J2) is a Juniper router. The Juniper operating system (JUNOS) basic configuration of BGP starts as it follows:

Router R2 (Juniper JUNOS)

```
J2#configure

set routing-options autonomous-system 2
set routing-options router-id 10.2.2.2
```

Figure 1.12

The autonomous system is declared in a global configuration mode because an ASBR can only be member of one autonomous system. Also, an identification address is assigned to the router; in this example, it is the IP address of a loopback interface.

In JUNOS, BGP Configuration Groups are created to peering with other AS. The idea is to place all the configuration commands in an organizational container that pertain to a particular BGP peering interest. This results in a very neat set up.

Router R2 (Juniper JUNOS)

```
J2#configure
set routing-options autonomous-system 2
set routing-options router-id 10.2.2.2
edit protocols bgp group EXAMPLE
        set type external
        set neighbor 10.0.12.1
        set peer-as 1
        set authentication-key EXAMPLE
        set export To_AS1
```

Router in configuration mode.

JUNOS requires the organization of BGP commands inside a group (sort of a container for BGP commands).

- This type is external BGP (because there is also an internal flavour of BGP).
- Neighbour address must be specified.
- The neighbour's AS number must also be included.
- An authentication password can be set.
- An export policy is being invoked here.

```
set policy-options policy-statement To_AS1 term 1 from route-filter 192.168.2.0/24 exact
set policy-options policy-statement To_AS1 term 1 then accept
```

This policy matches and accepts the IP prefix to be advertised.

Figure 1.13

These are the basic Juniper configuration tasks. However to export routing knowledge additional configuration has to be done. Inside the bgp group EXAMPLE, an export statement is created:

```
set export POLICY-NAME
```

This export statement relates to a policy that has to be written in JUNOS like this:

```
set policy-options policy-statement THE-POLICY-NAME-GOES-HERE from "something"
set policy-options policy-statement THE-POLICY-NAME-GOES-HERE then accept
```

Where "from something" can be many things. For example, the following statement accepts the IP route that is going to be advertised by matching its directly connected interface:

```
set policy-options policy-statement EXAMPLE from protocol direct interface em0.0
set policy-options policy-statement EXAMPLE then accept
```

But, alternatively, this will also work:

```
set policy-options policy-statement EXAMPLE from protocol direct interface 192.168.2.1
set policy-options policy-statement EXAMPLE then accept
```

Finally, the next option is the most precise and elegant. Since the route must be present in the IP routing table, then it makes sense to just match it with a route filter regardless of the protocol or interface of origin:

```
set policy-options policy-statement EXAMPLE from route-filter 192.168.2.0/24 exact
set policy-options policy-statement EXAMPLE then accept
```

In the previous explanation, the policy EXAMPLE has only one condition and one action. However, a policy typically has more statements to match other conditions, so it would be better to leave it prepared in case that more routes are going to be added later. To make lists of conditions, JUNOS conveniently uses the keyword TERM followed by a string descriptor, a name basically.

```
set policy-options policy-statement EXAMPLE term NAME from route-filter 192.168.2.0/24 exact
set policy-options policy-statement EXAMPLE term NAME then accept
```

For example, let's suppose that another route 172.16.2.0/24 is to be advertised.

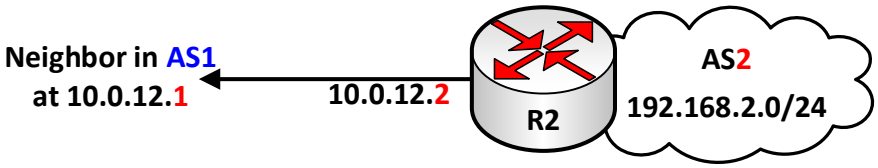
```
set policy-options policy-statement EXAMPLE term 1 from route-filter 192.168.2.0/24 exact
set policy-options policy-statement EXAMPLE term 1 then accept
set policy-options policy-statement EXAMPLE term 2 from route-filter 172.16.2.0/24 exact
set policy-options policy-statement EXAMPLE term 2 then accept
```

In this case the description used is a number because it is so simple to follow, but the descriptor could be anything such as letters, words, or phrases.

```
set policy-options policy-statement EXAMPLE term A from route-filter 192.168.2.0/24 exact
set policy-options policy-statement EXAMPLE term A then accept
set policy-options policy-statement EXAMPLE term B from route-filter 172.16.2.0/24 exact
set policy-options policy-statement EXAMPLE term B then accept
```

Beware that the numbers or letters are just names and they do not control the order of execution of the list but rather the terms are executed in the sequence in which they are written. For example, if term 1 is deleted and then added after term 2, then the policy executes term 2 first, and then term 1.

Finally, the complete BGP configuration (with comments) for a Juniper router looks like this:



Router R2 (Juniper JUNOS)

```
J2#configure
```

Router in configuration mode.

```
set interface lo0.0 family inet address 10.2.2.2/32
set interface em0.0 family inet address 192.168.2.1/24
set interface em1.0 family inet address 10.0.12.2/30
```

Configuration of local interfaces

```
set routing-options autonomous-system 2
set routing-options router-id 10.2.2.2
```

Set BGP autonomous system identification

Set router identification to the loopback address

```
edit protocols bgp group EXAMPLE
        set type external
        set neighbor 10.0.12.1
        set peer-as 1
        set authentication-key EXAMPLE
        set export To_AS1
```

This group contains all the configuration commands pertaining to BGP such as peering type, the IP address and ASN of the neighbour, the authentication key and export/import policies.

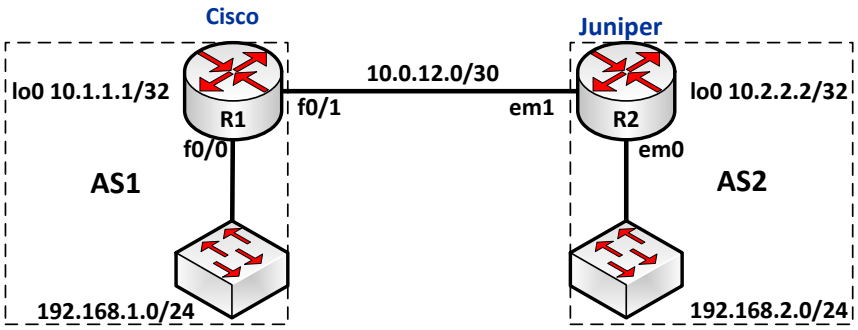
```
set policy-options policy-statement To_AS1 term 1 from route-filter 192.168.2.0/24 exact
set policy-options policy-statement To_AS1 term 1 then accept
```

To_ AS1 is a policy that matches and accepts the IP prefix to be advertised. BGP receives the prefix information as it is filtered by this policy.

Figure 1.14

Configuration Scenario for both Cisco IOS and Juniper

After covering the fundamental configuration tasks for both vendors, a simple network with only two routers will be used to compare the configuration for both operating systems Cisco IOS (left) and Juniper JUNOS (right).



Cisco Router R2

```
interface lo0
  ip address 10.1.1.1 255.255.255.255

interface f0/0
  ip address 192.168.1.1 255.255.255.0
  no shutdown

interface f0/1
  ip address 10.0.12.1 255.255.255.252
  no shutdown

router bgp 1
neighbor 10.0.12.2 remote-as 2
network 192.168.1.0 mask 255.255.255.0
```

Juniper Router R2

```
set interface lo0.0 family inet address 10.2.2.2/32
set interface em0.0 family inet address 192.168.2.1/24
set interface em1.0 family inet address 10.0.12.2/30

set routing-options autonomous-system 2
set routing-options router-id 10.2.2.2

edit protocols bgp group EXAMPLE
        set type external
        set neighbor 10.0.12.1
        set peer-as 1
        set export SEND_TO_AS1

edit policy-options policy-statement SEND_TO_AS1
        from route-filter 192.168.2.0/24 exact
        then accept
```

Figure 1.15

In Cisco IOS, the "network" command matches the local network prefix that is desired to be advertised. The keyword "mask" is to match exactly the address mask of the network that is to be advertised. For legacy reasons, former classful addresses do not need the keyword mask.

In Juniper JUNOS, the line command "export NAME" will call upon a number of orderly statements just like a sub-routine call in a program. In this example, the network prefix to be advertised is matched with a filter command hence the statement "from route-filter". In the example above, the initial configuration of Cisco IOS is obviously simpler. However, the Juniper

JUNOS configuration has already a format which facilitates the additional building of BGP policies. So, in the longer term, this initial configuration extra work pays off.

R1 Routing Table

In Cisco IOS, the command "show ip bgp" indicates the IP prefixes present in a BGP database. These prefixes will be presented to the IP protocol and IP will choose the ones that will be installed in the IP routing table. In this case, R1 shows one prefix, 192.168.1.0/24 present in the database with a next hop of 0.0.0.0 and a path of i or internal to this domain. That means that R1 has the prefix ready for advertisement to a neighbour. The other prefix 192.168.2.0/24 is being advertised via the next hop 10.0.12.2 with an AS (autonomous system) path of 2 so this is the routing knowledge that is being advertised, via BGP, by the Juniper router R2.

```
R1#show ip bgp
router ID is 10.1.1.1
   Network        Next Hop       Metric  LocPrf  Weight   Path
*> 192.168.1.0    0.0.0.0        0                32768    i
*> 192.168.2.0    10.0.12.2      0                 0       2  i
```

Table 1.1

This is the IP routing table in router Cisco R1. The route 192.168.2.0/24 (in bold) was advertised by R2 using BGP. The number **20** is the Cisco **Administrative Distance (AD)** to indicate a level of preference for this route. In principle, the same route could be presented to the IP protocol from different protocols sources, for example OSPF. Routes learned via External BGP are assigned an AD of 20 by default in a Cisco router while OSPF is 110. The lowest the AD value, the more preferred, so BGP beats OSPF if both protocols present the same routing information.

```
R1#show ip route
C     10.0.12.0/30 is directly connected, f0/1
C     10.1.1.1/32 is directly connected, Lo0
C   192.168.1.0/24 is directly connected, f0/0
B   192.168.2.0/24 [20/0] via 10.0.12.2
```

Table 1.2

R2 Routing Table

In Juniper, the command "show route" indicates the knowledge about prefixes but from the perspective of R2. The command "show route terse" provides a more organized view of the routing table. Router R2 shows one routing table called inet.0. (in JUNOS, **inet** stands for IPv4).

felix@R2> **show route terse**

IPv4 routing table (inet.0): * active destinations

A	Destination	Protocol	Preference	Metric 1	Metric 2	Next hop	AS-path
*	10.2.2.2/32	Direct (D)	0			lo.0	
*	10.0.12.0/30	Direct (D)	0			em1.0	
*****	**192.168.1.0/24**	**BGP (B)**	**170**	**100**	**0**	**10.0.12.1**	**1 i**
*	192.168.2.0/24	Direct (D)	0			em0.0	

Table 1.3

BGP Basic Conversation

Once the two routers are configured to speak BGP, they will have a logical conversation. The figure shows the initial steps of this BGP conversation.

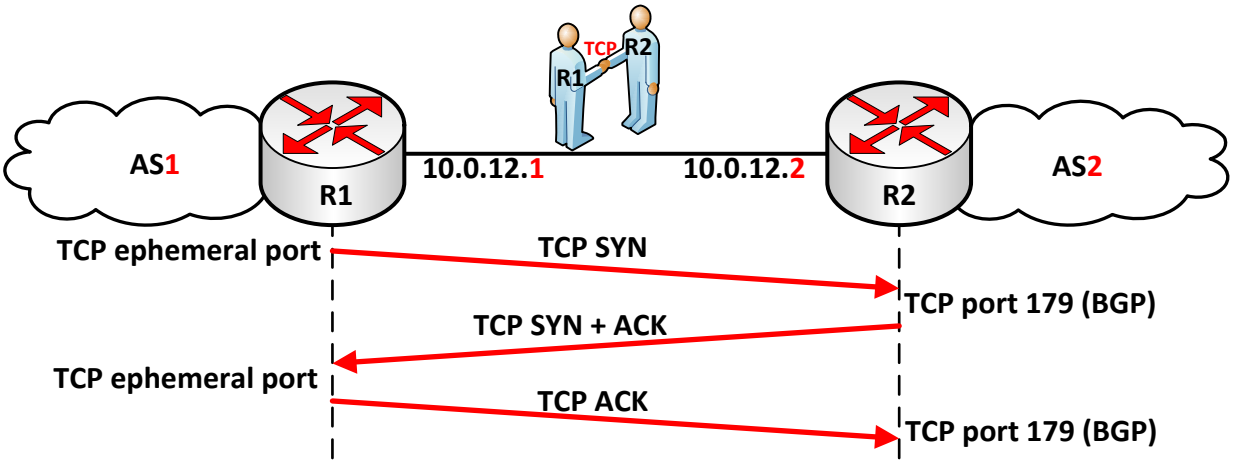


Figure 1.16

A TCP session must be started first since TCP is the transport protocol for BGP messages. One of the routers attempts to start such TCP session with a TCP **SYN**chronization message. In this diagram, router R1 sources the session from an ephemeral port to the BGP destination port number which is $179_{(decimal)}$. Router R2 responds with its own SYN to open the session from the other way. Both opening TCP messages are (**ACK**) acknowledged back. Once that the two routers successfully achieve a three-way TCP handshake, they are ready to open the BGP communication session by exchanging BGP OPEN messages.

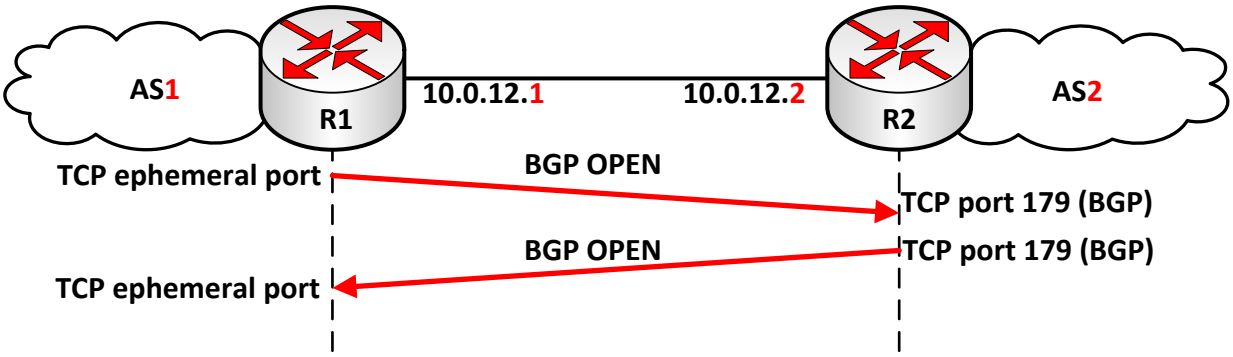


Figure 1.17

The OPEN messages act as presentation cards for the routers. During this OPEN stage, the routers are looking at each other BGP features to see if they can have a compatible conversation. The two routers would refuse to talk if the descriptions in the OPEN message would not match those in the configuration parameters or if the routers would not support a major capability. In that were to happen, they would exchange NOTIFICATION messages to explain why they could not talk. **BGP NOTIFICATION** messages contain **error codes**.

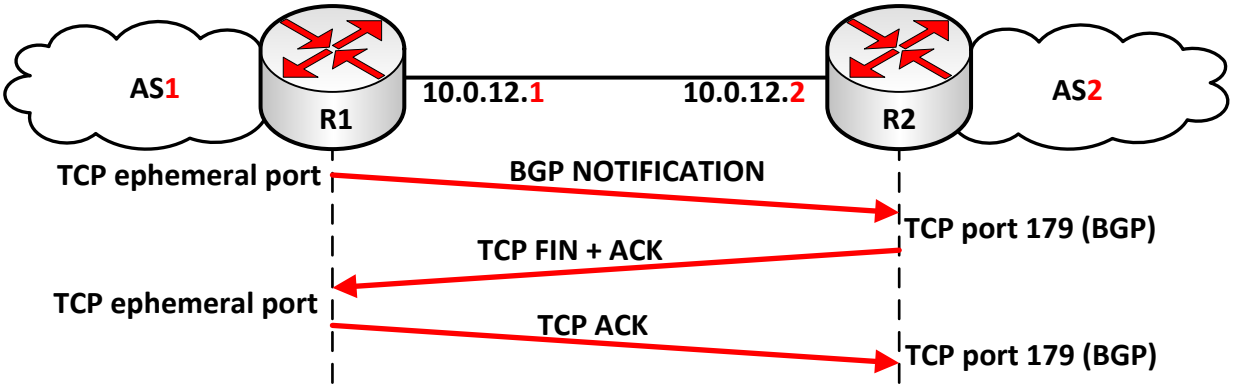


Figure 1.18

If the routers find themselves to be compatible to carry out a conversation, they proceed to exchange BGP UPDATES. These messages contain the actual routing information.

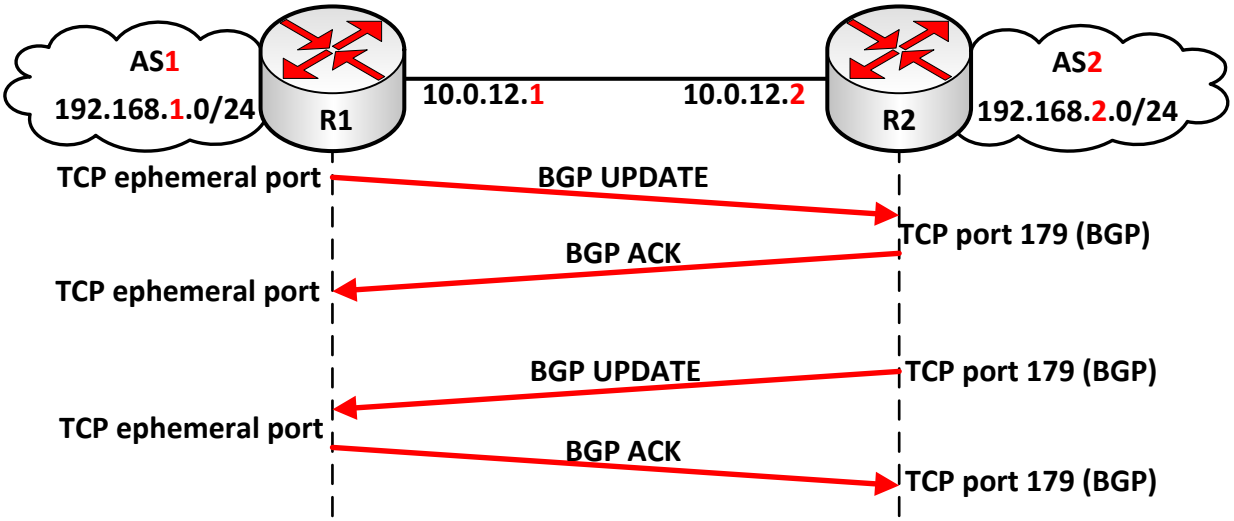


Figure 1.19

In the present case, router R1 has only one prefix to advertise to R2 initially and vice versa. The diagram below shows router R1 sending a BGP UPDATE message with the complete layered encapsulation. Once the two routers exchange UPDATES, they know how to reach the external IP routes.

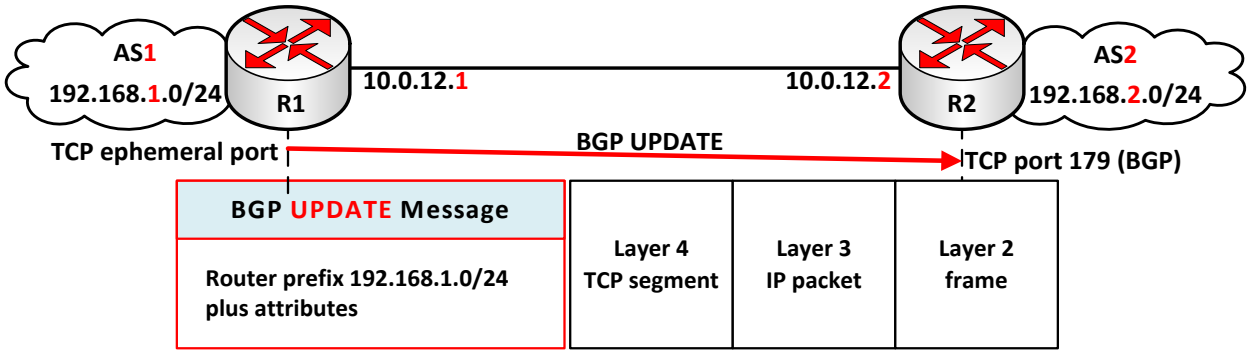


Figure 1.20

After the two routers have completed their exchange of routing information, they have nothing else to say unless something happens like a network going down. In an event like that, another UPDATE is issued, but this time to withdraw the now unreachable prefix from the peer routers. If nothing goes up or down, the two routers remain mostly quiet, sending each other BGP headers that work as **BGP KEEPALIVE** messages. These KEEPALIVES are also acknowledged reciprocally. Hence, in a very stable network topology, an observer of BGP traffic will only see KEEPALIVES and TCP ACKs being exchanged periodically.

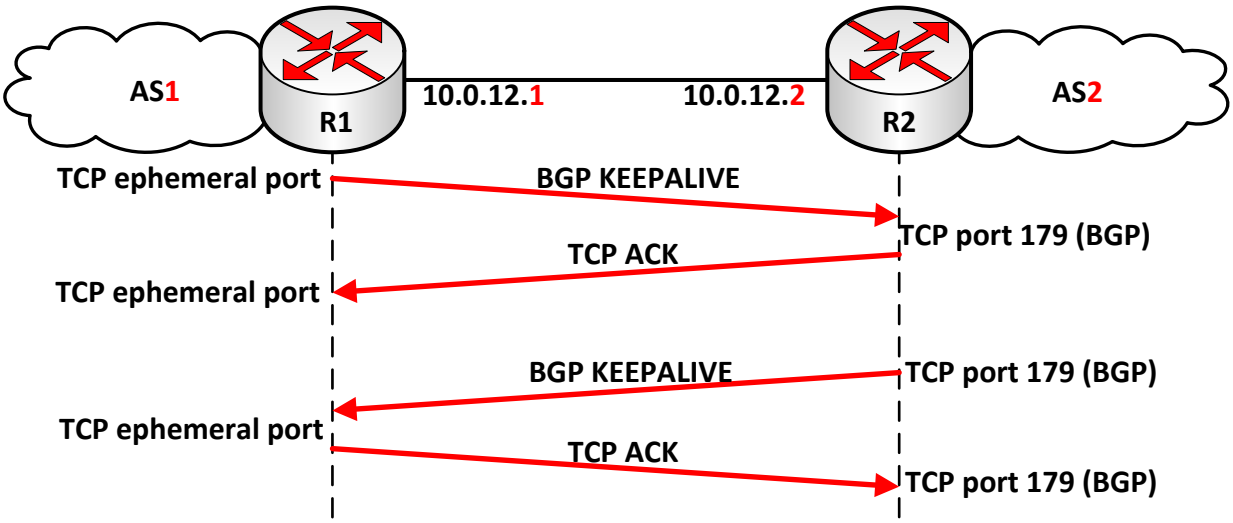


Figure 1.21

Configuration of BGP to carry IPv6 routing information

Internet Protocol version 6 (IPv6) is actually a different protocol than IPv4. Basically, the only things that they have in common is the IP name and that they are both layer 3 network protocols; but apart from that, they are not even compatible. Nevertheless, the information about IPv6 routes needs to be propagated in the same way than in IPv4, and BGP is the most capable protocol to do just that.

A simple network topology will be used to demonstrate the configuration of BGP to carry IPv6 routes using both Cisco IOS and Juniper routers. The whole topology is configured with IPv6 addresses only except for the loopback IPv4 addresses which are being used to provide identity to the routers. The configuration of the Cisco router R1 is explained in the next diagram.

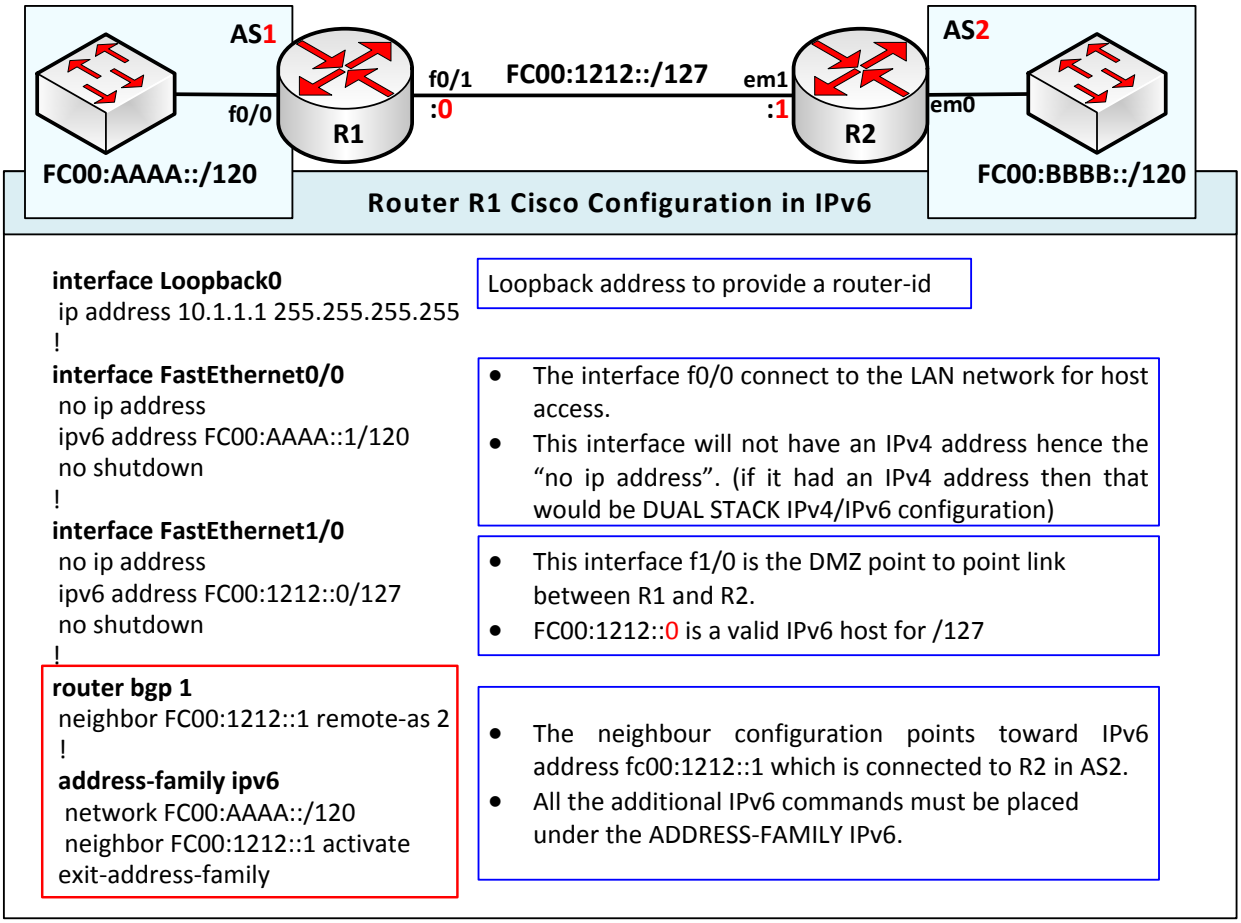


Figure 1.21

Beside the IPv6 addresses, the configuration of BGP is also slightly different. The major departure from IPv4 is that the injection of the IPv6 routes is done under an "**address-family**" which contains the definition of the protocol being advertised. In short, BGP can carry IPv6 routing information because it supports **multiple protocols**. The configuration of the Juniper router is explained in the next diagram.

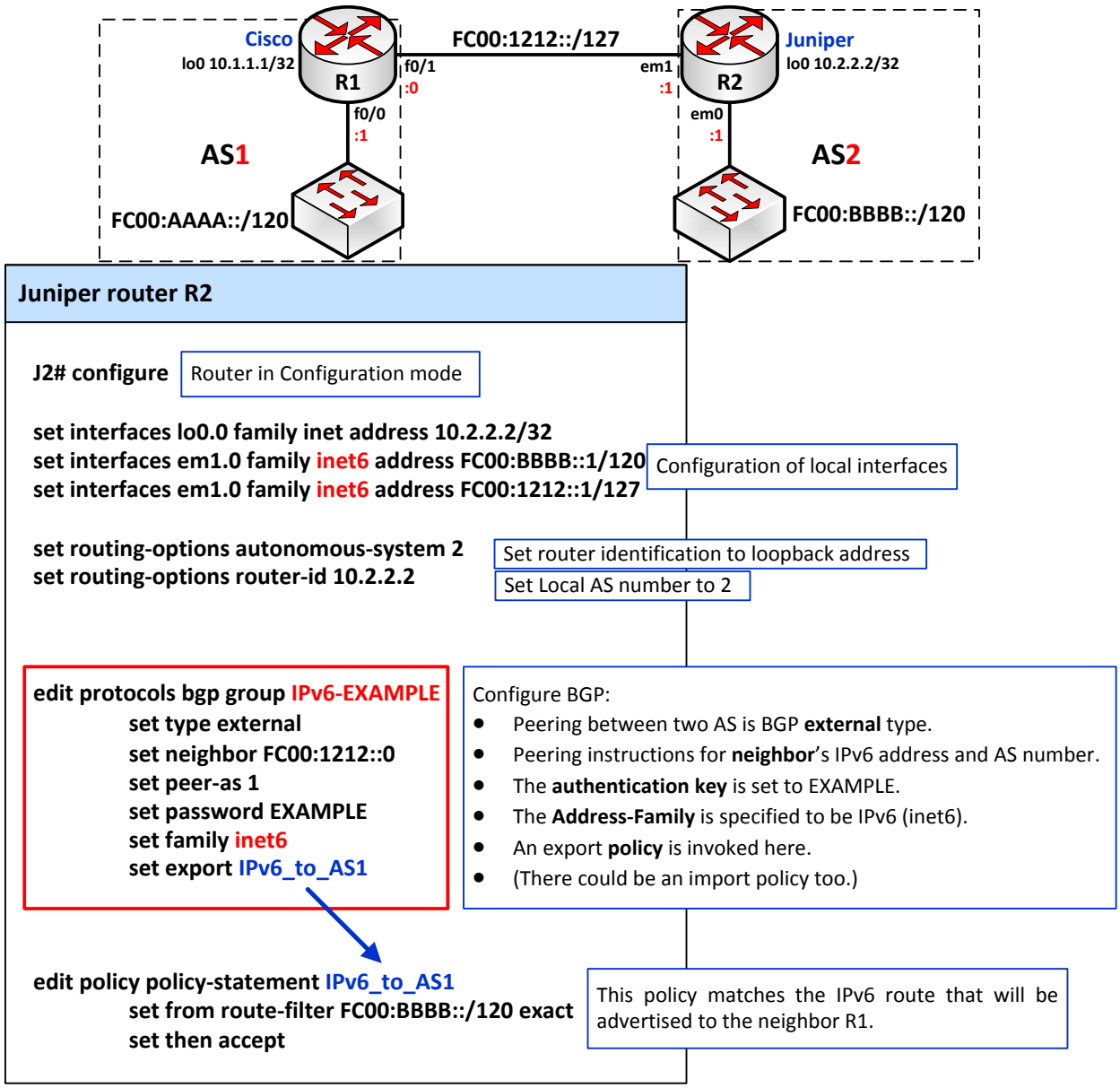


Figure 1.22

As soon as the configuration is completed the two routers start a BGP conversation with a three way TCP handshake, then the two routers exchange OPEN messages. In these OPEN messages, the routers advertise their capabilities. The following OPEN message indicates that R2

understands IPv6 routing information by using the **Multiprotocol Capability Address-family IPv6 Unicast** (AFI code number 2, sub-code SAFI number 1).

Border Gateway Protocol
Marker FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
Length: 45 bytes
TYPE: (number 1) OPEN Message

My BGP version is 4
My Autonomous System Number is 2
My Hold Time is 90 seconds
My BGP identifier is 10.2.2.2
Optional Parameters Length = 16 bytes
Optional parameters:
Optional parameter: Capability (2)
Parameter Length = 6 Bytes
Capability: Multiprotocol Extensions Capability
Multiprotocol Extensions Capability Type: (1)
Length = 4 bytes
Address Family Identifier AFI number 2 = IPv6
Reserved: 00
Subsequent AFI (SAFI) = 1 or Unicast
Optional parameter: Capability
Optional parameter: Capability

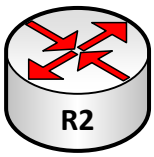


Figure 1.23

Router R1 will do the same and advertise its multiprotocol IPv6 support. Each router reviews the received OPEN and since both are properly configured to understand IPv6, they proceed to exchange UPDATE messages. If the routers would find a major incompatible feature, then an ERROR NOTIFICATION Message would be sent instead.

This is a BGP UPDATE message sourced from R1. The IPv6 routing information is contained inside the field Path Attribute Multi-Protocol Reachability Network Layer Information (MP_REACH_NLRI).

Border Gateway Protocol
Marker FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
Length: 97 bytes
TYPE: (number 2) UPDATE Message

Withdrawn Routes Length = 0
Total Path Attribute Length = 74 bytes
Path Attributes:
  MP_REACH_NLRI
  ORIGIN = IGP
  AS_PATH = 1
  MULTI_EXIT_DISCRIMINATOR MED = 0

L4 Transmission Control Protocol: Source Port **179 (BGP)** Destination Port 55307
L3 Internet Protocol version 6: Source fc00:1212::0 Destination fc00:1212::1
L2 Ethernet Protocol: Destination MAC 00:00:ab:af:94:01 Source ca:01:13:18:00:1c

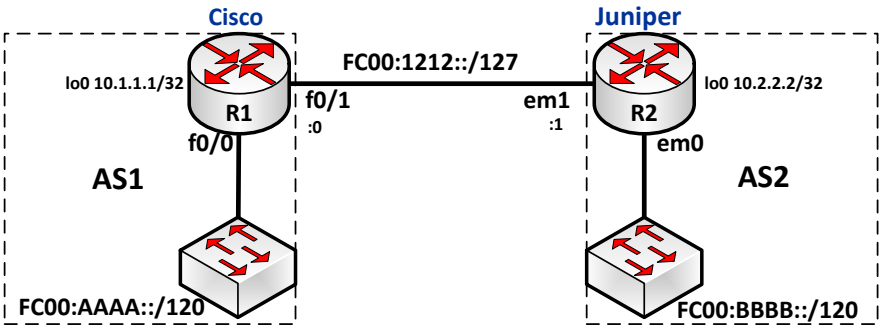
Figure 1.24

In effect, the MP_REACH_NLRI attribute (code 14) contains the IPv6 prefix FC00:AAAA:: with the mask /120. This is how BGP propagates the IPv6 routing knowledge. Here are the details.

Path Attribute: MP_REACH_NLRI
Flags 0x80 Type Code 14 Length 53 bytes
Address Family Identifier **AFI (2) IPv6** SAFI (1) Unicast
Next Hop Address 32 bytes
Network Layer Reachability Information (16 Bytes)
FC00:AAAA::/120
  MP Reach NLRI prefix length = 120 bits
  MP Reach NLRI IPv6 prefix **FC00:AAAA::**

Figure 1.25

Finally, the routing tables of both routers show the routing information complete. A ping from R1 to FC00:BBBB::1 and from R2 to FC00:AAAA::1 confirms that IPv6 packets can be delivered.



Cisco Router R1 routing table

R1#sh ipv6 route

C FC00:AAAA::/120 [0/0]
via f0/0, directly connected

B FC00:BBBB::/120 [20/0]
via FC00:1212::1

C FC00:1212::/127 [0/0]
via f1/0, directly connected

Juniper Router R2 routing table

felix> show route terse

A Destination	P	Prf	Metric 1	Next hop	AS path
*** fc00:AAAA::/120**	**B**	**170**	**100**	**> fc00:1212::**	**1 I**
* fc00:BBBB::/120	D	0		>em0.0	
* fc00:1212::/127	D	0		>em1.0	

```
R1#ping fc00:bbbb::1
reply 100%
```

```
R2#ping fc00:aaaa::1
reply 100%
```

Figure 1.26

An Overview of BGP in the Internet Context

Judging by the previous examples, BGP does not appear to be a particularly complex protocol, but that is only in its basic mode because BGP configuration gets very complex when it is implemented by Internet providers and carriers.

BGP is different to all the other routing protocols. For any of the IGP protocols (OSPF, ISIS, EIGRP, and RIP) the administrator just needs to type the configuration commands and let the protocol works its ways according to a particular algorithm (Dijkstra or Garcia Luna or Bellman-Ford-Fulkerson).

In BGP, the administrator must configure constrains and policies that influence the path determination. BGP is designed to allow the humans to manipulate the Internet traffic. The path that the IP traffic is going to follow is determined by contractual and commercial pacts and BGP is a tool to implement just that. It could be said that the BGP protocol is completed by the human who writes the policies. A look into actual autonomous systems illustrates this basic BGP behaviour. This is according to data [10] from May 2016 with the following disclaimers:

- All this information is in the public domain and it can be obtained with the right searching tools.
- This information is being used for educational purposes only.
- The information might have changed by the time that this document is read.

The starting point is the Autonomous System AS 5664 belonging to Sheridan College located in Ontario, Canada. A picture of clouds better represents its relationships:

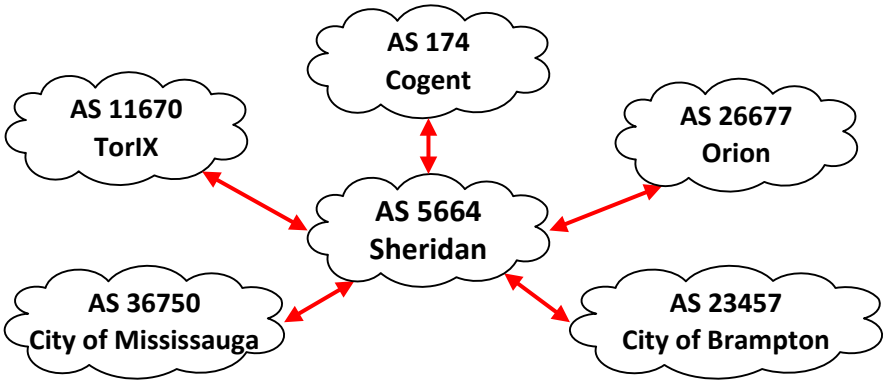


Figure 1.27

This is the general interpretation of this overview topology:

- Sheridan College owns the IPv4 address space 142.55.0.0/16 and the Autonomous System number 5664. This address range contains (2^{16}) 65,536 IPv4 addresses. Sheridan can advertise this network prefix or subsets of the space.

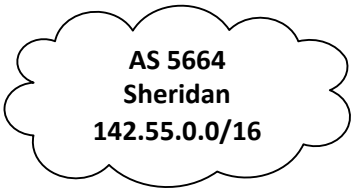


Figure 1.28

- Sheridan receives Internet service from provider Cogent AS 174. The bulk of the Internet traffic inbound and outbound can go this way.

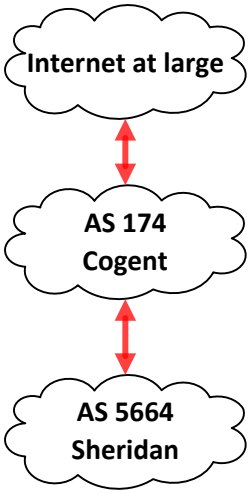


Figure 1.29

- Sheridan also peers with TorIX AS 11670 which is the Toronto Internet Exchange. Sheridan can also obtain access to the rest of the Internet via this path. The network administrator must set up policies to use the two paths available, Torix and Cogent, for inbound and outbound traffic.

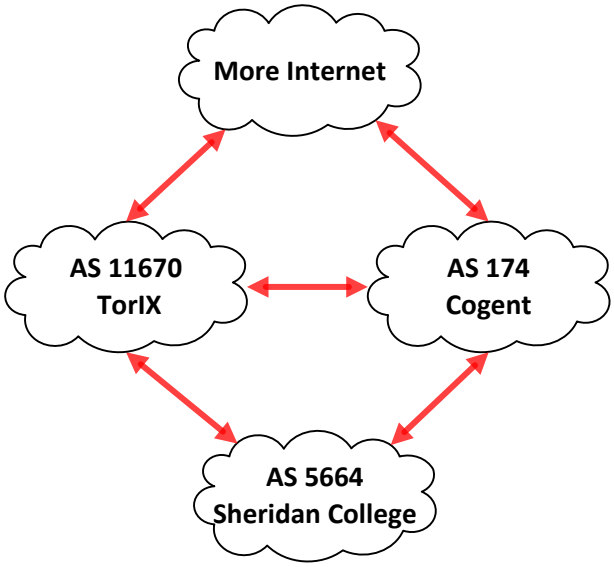


Figure 1.30

- In the Toronto Internet Exchange [14] building, Sheridan can potentially find hundreds of other AS. Private peering can be established where two AS partners agree to mutually show their local routing information or parts of that. If this is the case, TorIX would just provide the physical infrastructure to make the interconnections.

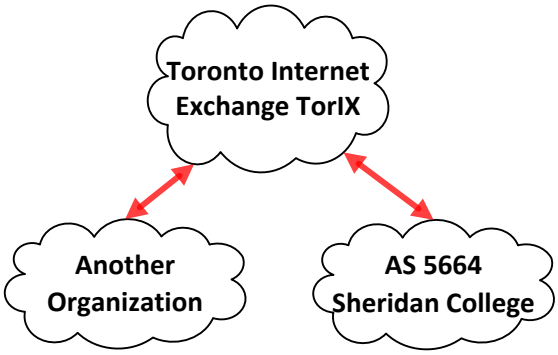


Figure 1.31

- Additionally, Sheridan peers with Orion AS 26677 but it does not get Internet access that way. Instead, Orion is a research network that interconnects colleges, universities, research centres, libraries and other public institutions in Ontario so Sheridan can set up policies to reach other similar educational institutions thru Orion.

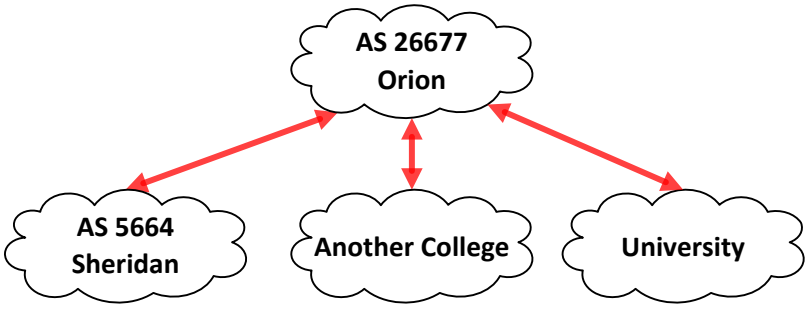


Figure 1.31

Finally, Sheridan provides transit to parts of the networks of the city of Mississauga and the city of Brampton as part of collaboration agreements.

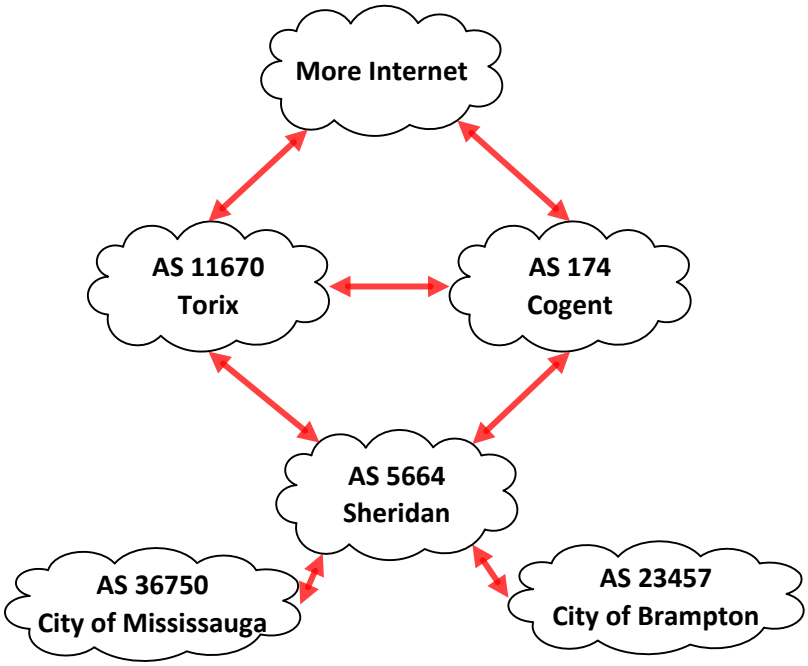


Figure 1.32

For every single case mentioned above, the administrator at Sheridan College must design and implement routing policies. Of course, to implement any scheme, there has to be agreements between the College and its peers. The intentions of such agreements, on paper, are actually implemented with BGP policies in practice. The BGP speaking routers behave according to the instructions given by the policies.

BGP can do that because it can tag or mark the routes with attributes. For BGP, IP prefixes are objects with properties. The policies implemented in the routers act upon those attributes. For instance, if the administrator at Sheridan would like to avoid transit from Orion to Cogent and vice versa going thru the College, then the administrator would configure a policy in the routers to filter the external advertisements and limit them to local routes only (from 142.55.0.0 /16).

The most basic BGP configuration just requires the specification of a neighbour router and the AS number to establish connectivity. But to actually control the network traffic with BGP, the configuration must include policies which are applied to different flow interest either inbound or outbound to the AS. This diagram gives a general idea of this principle in pseudo-code for both Cisco and Juniper.

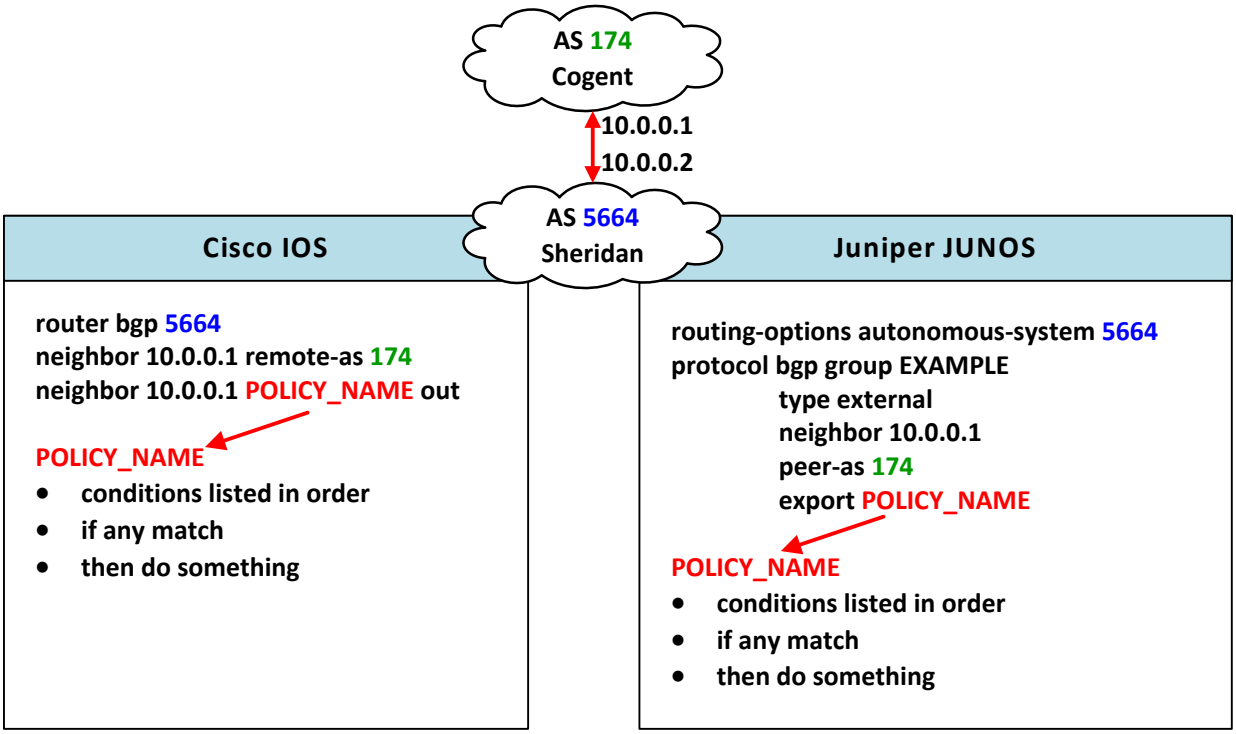


Figure 1.33

All routing advertisements destined to AS174 will have to be validated against the policy conditions. The contents of the BGP UPDATE message will be the result of passes thru the conditions set up by the administrator. There are several techniques to build these policies such as route-maps, filter-lists, prefix-lists, access-lists, policy statements, policy chains and others.

The bulk of the work in BGP configurations is the design, deployment and maintenance of the routing policies. The intent of the policies depends on the function and position of the autonomous systems in the Internet. Some domains provide access to other AS, others provide transit, and others are just customers of service providers.

The relationship among Internet Service Providers (ISP) and customers; and among network providers themselves is quite complex. For example, a general schematics is shown next using collected [10, 11, 12] information. The AS border routers are not shown in the figure because the actual details of the exact placement and connections are unknown to the author. Also, not all connections and autonomous systems are portrayed because that would not be feasible anyways. In general terms, these clouds interconnect like this:

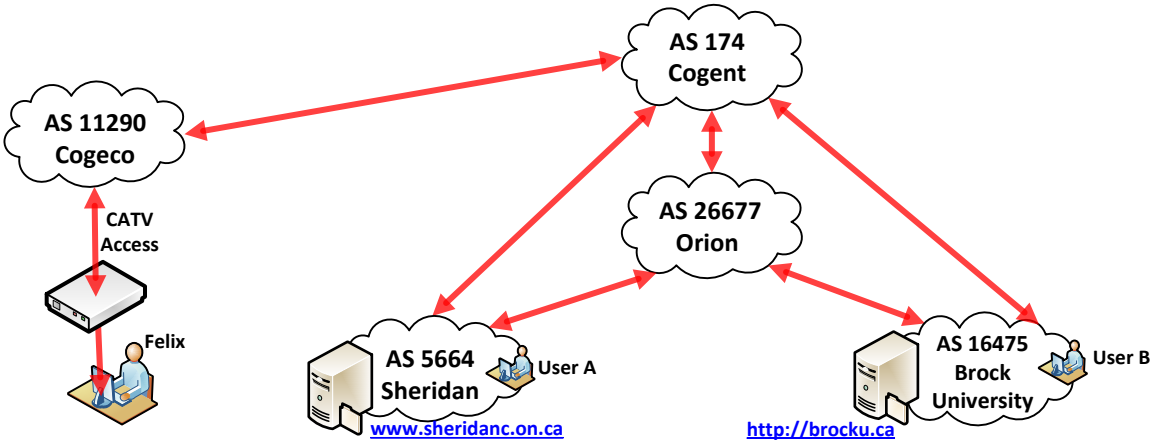


Figure 1.34

Organizations such as Sheridan and Brock University can configure collaboration networks via Orion. This network provides connectivity to dozens of client networks, most of them academic and research networks in Ontario. For example, the traffic between the generic User A located inside Sheridan and User B located in a network in Brock University could go thru the Orion depending of the policies in both sites. In turn, Orion peers with other networks such as Canarie,

Cogent and Hurricane Electric HE (not shown in the diagram). But neither Sheridan nor Brock prefer to access the Internet via AS26677 because that is not the function of Orion. Instead, they would prefer (policies again) to use Cogent to reach the rest of the Internet.

Cogent [11] has a gigantic network that connects with thousands of other autonomous systems in the Internet. One of them is LEVEL 3 Communications AS 3356 [12] also with several thousand adjacencies. LEVEL 3 peers with AS 7018 which belongs to AT&T.

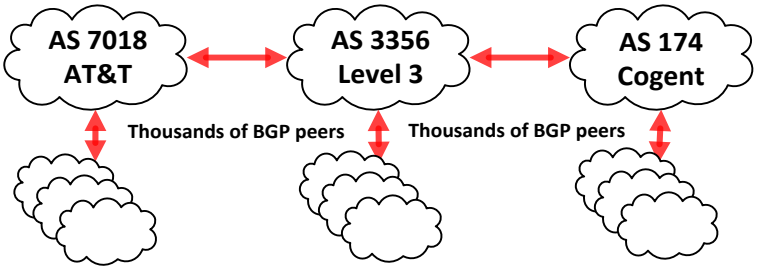


Figure 1.35

These are the true giants of the Internet with networks that spawn throughout continents and infrastructures that require several thousand routers in place. Just imagine the maintenance and configuration tasks done on a daily basis in these companies!

Finally, this panorama is completed with an exploration of the traffic sent from a user named Felix to Sheridan College website www.sheridanc.on.ca. This user receives Internet home access from Cogeco, a CATV provider. In turn, Cogeco with the AS number 11290, peers with Cogent so the traffic between the two end locations flows thru autonomous systems AS11290, AS174, and AS5664. However, there could potentially be other paths that BGP can choose based on the examination of the attributes and policies.

This partial picture of just one corner of the Internet gives an idea of its complexity. From BGP viewpoint though, the Internet is the interweaving of clouds that contain network prefixes to which attributes are assigned. Those attributes are used to make routing decisions.

The Internet is this amazing mesh of autonomous systems and the glue that keeps them connected is BGP!

Chapter Exercises

Practice Basic BGP

- Implement the following network topology:

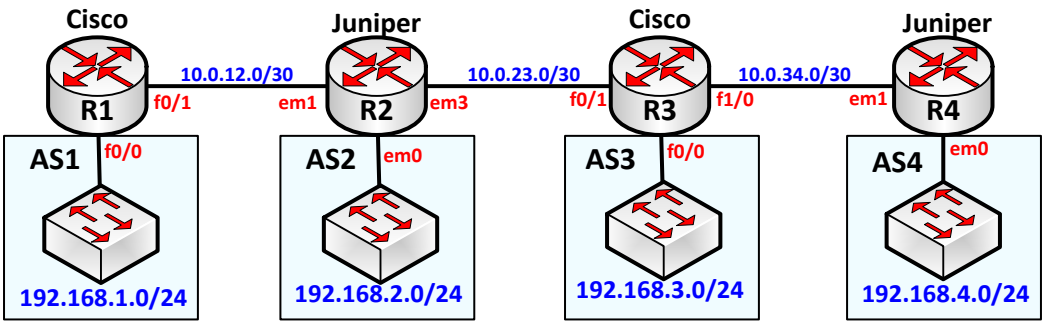
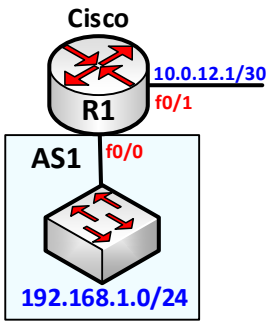


Figure 1.36

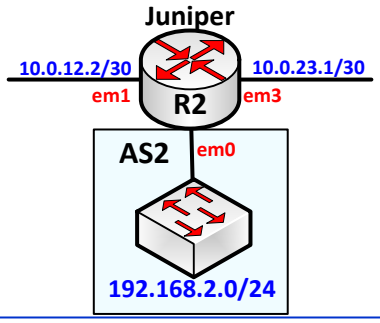
- Configure BGP between the Autonomous Systems according to the diagram.
- Once the configuration is complete verify that the knowledge about the 4 LAN networks appear in the routing tables of all the routers.
- Additional tasks:
    - ping from R1 the destination 192.168.4.1
    - ping from R4 the destination 192.168.1.1
    - ping from R1 to the address 10.0.34.2 (think about this, sniff the ping, analyze it)
    - ping from R1 to the address 10.0.23.1 (think about this, sniff the ping, analyze it)
- In Cisco routers, try "show ip bgp".
- In Juniper routers, try "show route protocol bgp".
- Attached is the reference configuration router by router.




```
hostname R1

interface lo0
ip address 10.1.1.1 255.255.255.255

interface f0/0
ip address 192.168.1.1 255.255.255.0
no shutdown

interface f0/1
ip address 10.0.12.1 255.255.255.252
no shutdown
```

```
router bgp 1
neighbor 10.0.12.2 remote-as 2
network 192.168.1.0
```

Figure 1.37

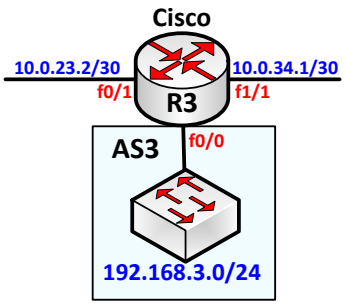



```
set system hostname R2
set interfaces lo0 unit 0 family inet address 10.2.2.2/32
set interfaces em0 unit 0 family inet address 192.168.2.1/24
set interfaces em1 unit 0 family inet address 10.0.12.2/30
set interfaces em3 unit 0 family inet address 10.0.23.1/30

set routing-options router-id 10.2.2.2
set routing-options autonomous-system 2
```

```
set protocols bgp group my_PEERS type external
set protocols bgp group my_PEERS neighbor 10.0.12.1 peer-as 1
set protocols bgp group my_PEERS neighbor 10.0.23.2 peer-as 3
set protocols bgp group my_PEERS export POLICY_OUT
```

```
set policy-options policy-statement POLICY_OUT term 1 from route-filter 192.168.2.0/24 exact
set policy-options policy-statement POLICY_OUT term 1 then accept
set policy-options policy-statement POLICY_OUT term 2 from protocol bgp
set policy-options policy-statement POLICY_OUT term 2 then accept
```

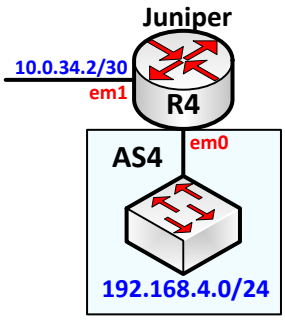
Figure 1.38




```
hostname R3

interface lo0
ip address 10.3.3.3 255.255.255.255

interface f0/0
ip address 192.168.3.1 255.255.255.0
no shutdown

interface f0/1
ip address 10.0.23.2 255.255.255.252
no shutdown
```

```
interface f1/0
ip address 10.0.34.1 255.255.255.252
no shutdown

router bgp 3
neighbor 10.0.23.1 remote-as 2
neighbor 10.0.34.2 remote-as 4
network 192.168.3.0
```

Figure 1.39




```
set system hostname R4

 set interfaces lo0 unit 0 family inet address 10.4.4.4/32
set interfaces em0 unit 0 family inet address 192.168.4.1/24
set interfaces em1 unit 0 family inet address 10.0.34.2/30

set routing-options router-id 10.4.4.4
set routing-options autonomous-system 4
```

```
set protocols bgp group to_AS3 type external
set protocols bgp group to_AS3 neighbor 10.0.34.1 peer-as 3
set protocols bgp group to_AS3 export POLICY_to_AS3_out
```

```
set policy-options policy-statement POLICY_to_AS3_out term 1 from route-filter 192.168.4.0/24 exact
set policy-options policy-statement POLICY_to_AS3_out term 1 then accept
```

Figure 1.40

Practice Basic BGP

- Configure basic BGP in the following network topologies:
- Implement with both Cisco and Juniper routers.
- Verify your IPv4 and IPv6 routing table.
    - Cisco IOS Command **show ip route** and **show ipv6 route**.
    - JUNOS Command **show route terse**.
- Verify and observe your BGP table.
    - Cisco IOS Command **show ip bgp** and **show bgp ipv6 unicast**.
    - JUNOS Command **show route protocol bgp terse**.

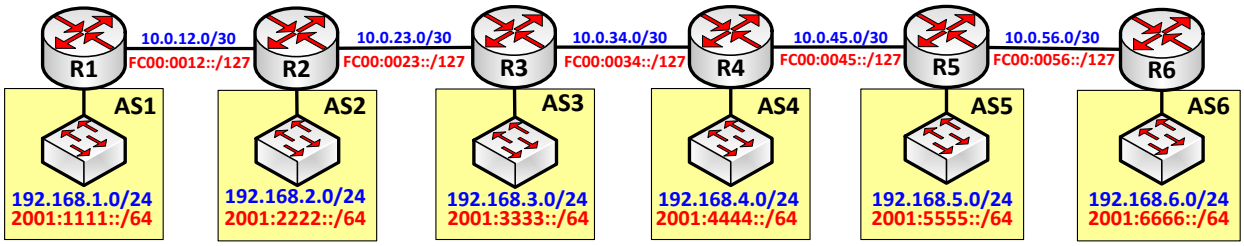


Figure 1.41

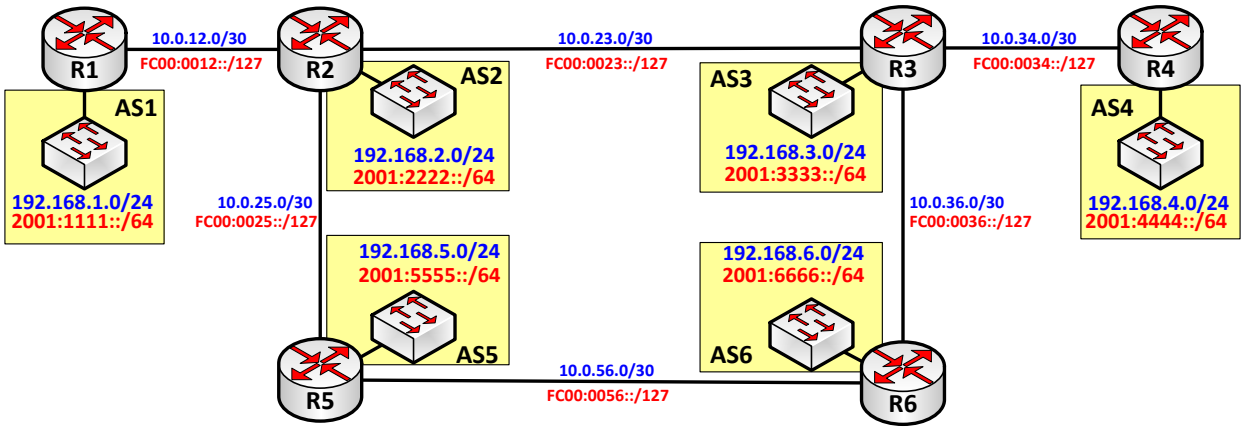


Figure 1.42

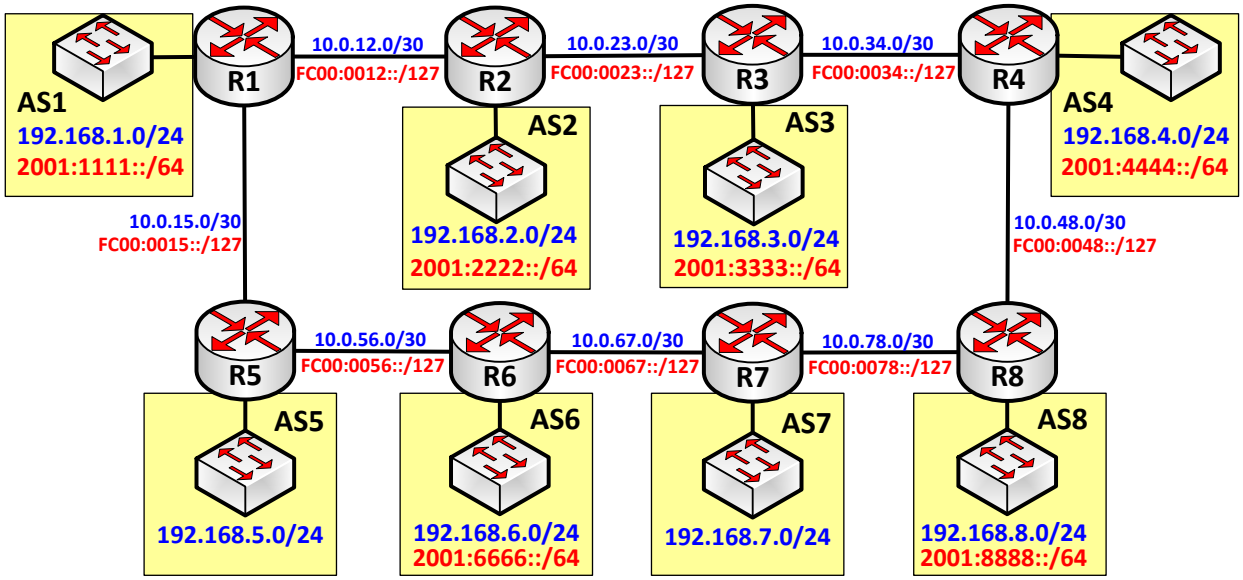


Figure 1.43

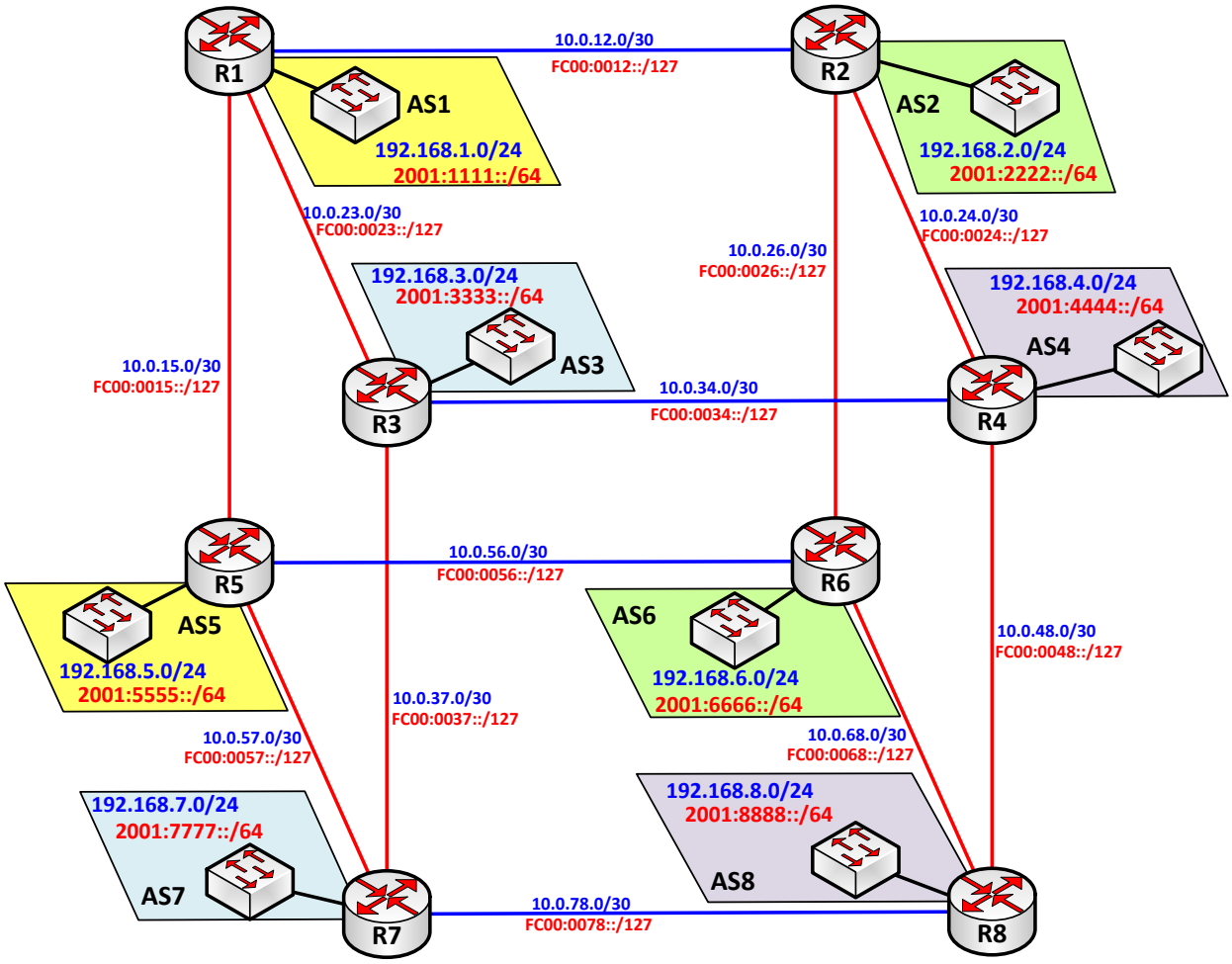


Figure 1.44

Labs

- Implement the "cylindrical" topology shown in the diagram.
- Router peering on the point to point links are: R1-R5, R2-R6, R3-R7, and R4-R8.
- These are the peer relationships over the DMZ-1 switch: R1-R2, R2-R3, R3-R4, and R4-R1.
- These are the peer relationships over the DMZ-2 switch: R5-R6, R6-R7, R7-R8, and R8-R5.
- Collect the information of the IP routing table and the BGP database in router R1.
- Using the IP routing table draw the map of IP connectivity from the viewpoint of R1.
- Using the BGP table draw the map of BGP knowledge from the viewpoint of Router R1.

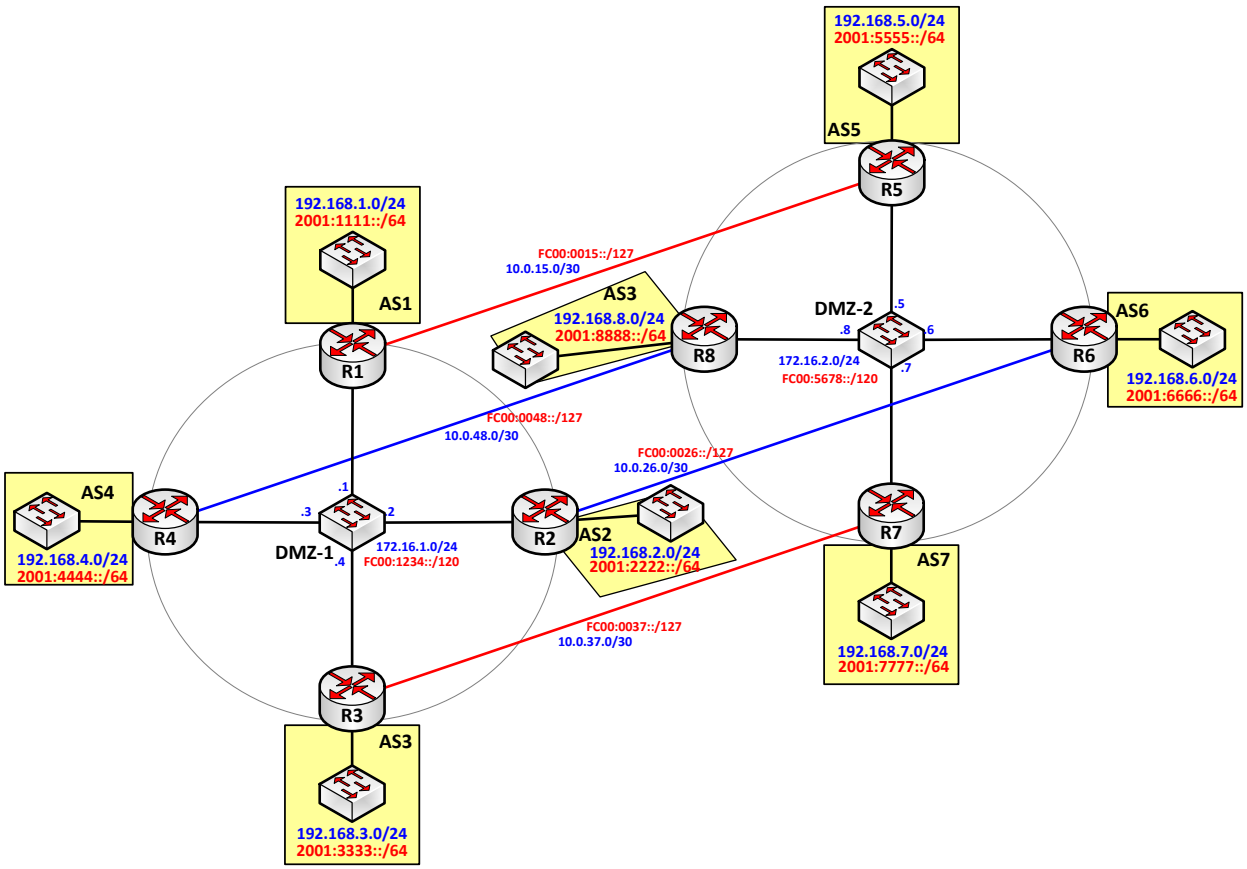


Figure 1.45

- Implement the "prism" topology shown in the diagram. Implement BGP as required.
- Using the IP routing table draw the map of IP connectivity from the viewpoint of R1.
- Using the BGP table draw the map of BGP knowledge from the viewpoint of Router R1.

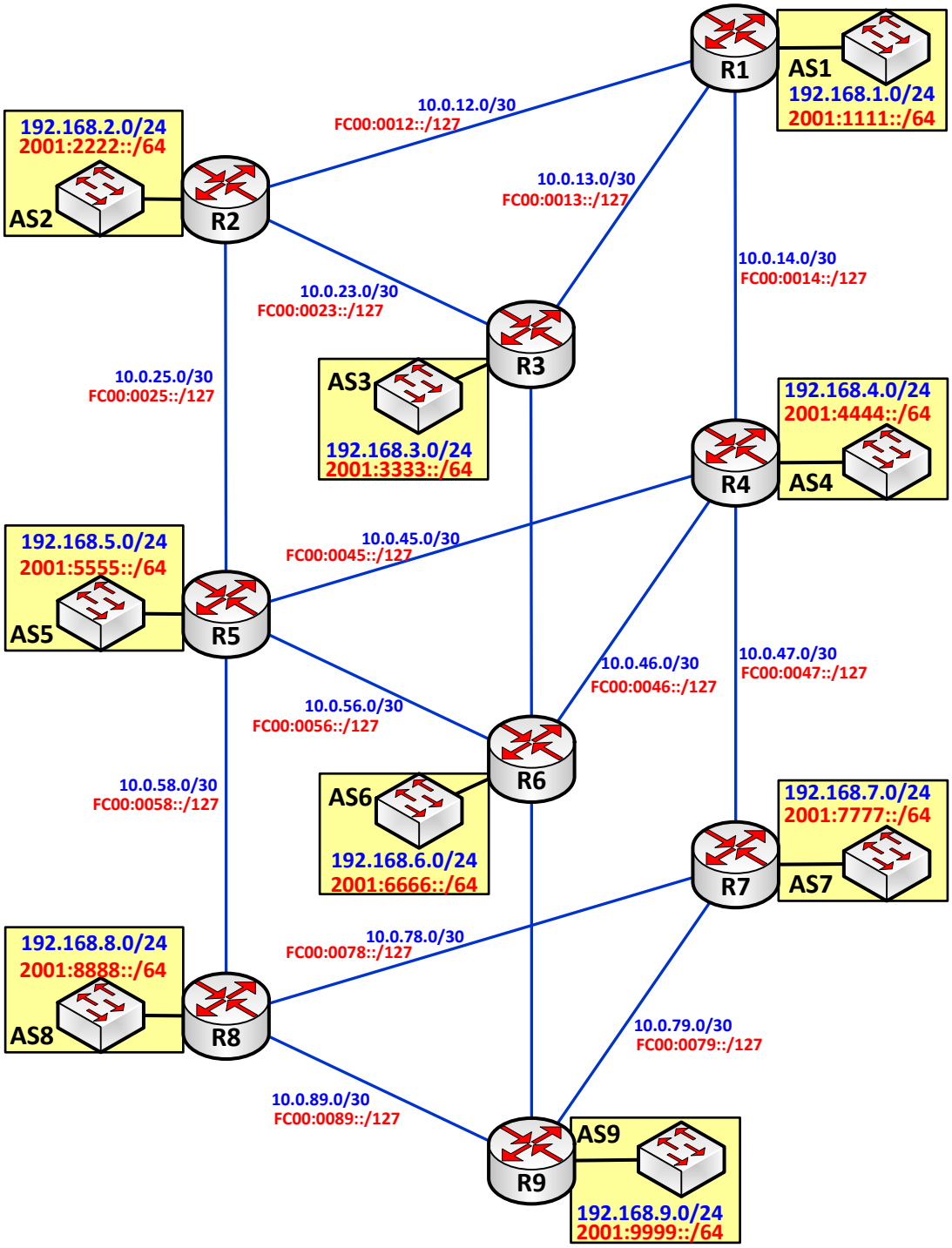


Figure 1.46

- Implement the "cubic" topology shown in the diagram. Implement BGP as required.
- Collect the information of the IP routing table and the BGP database in router R1.
- Using the IP routing table draw the map of IP connectivity from the viewpoint of R1.
- Using the BGP table draw the map of BGP knowledge from the viewpoint of Router R1.

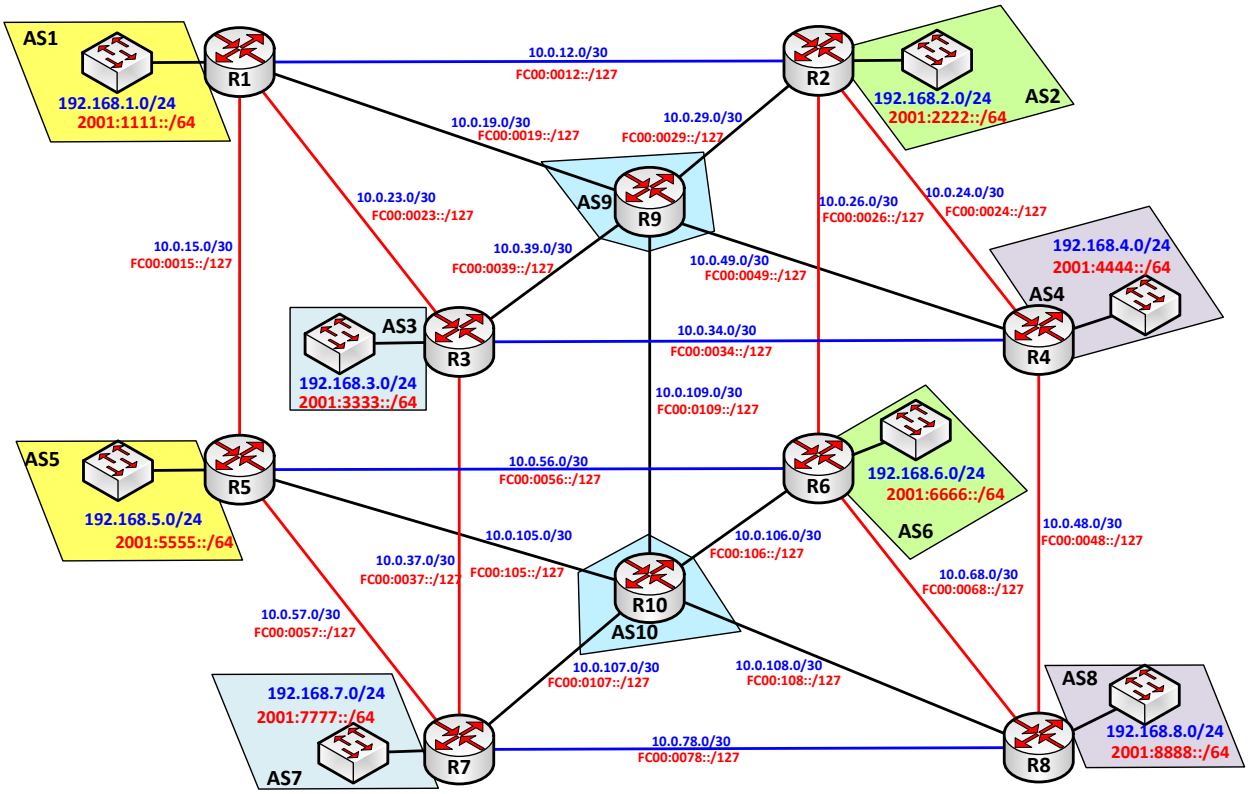


Figure 1.47

- Implement the following topology and set up BGP adjacencies. Verify that every router have reachability to all network destinations in both stacks IPv4 and IPv6. From information collected draw the maps of IP connectivity and BGP topological knowledge.

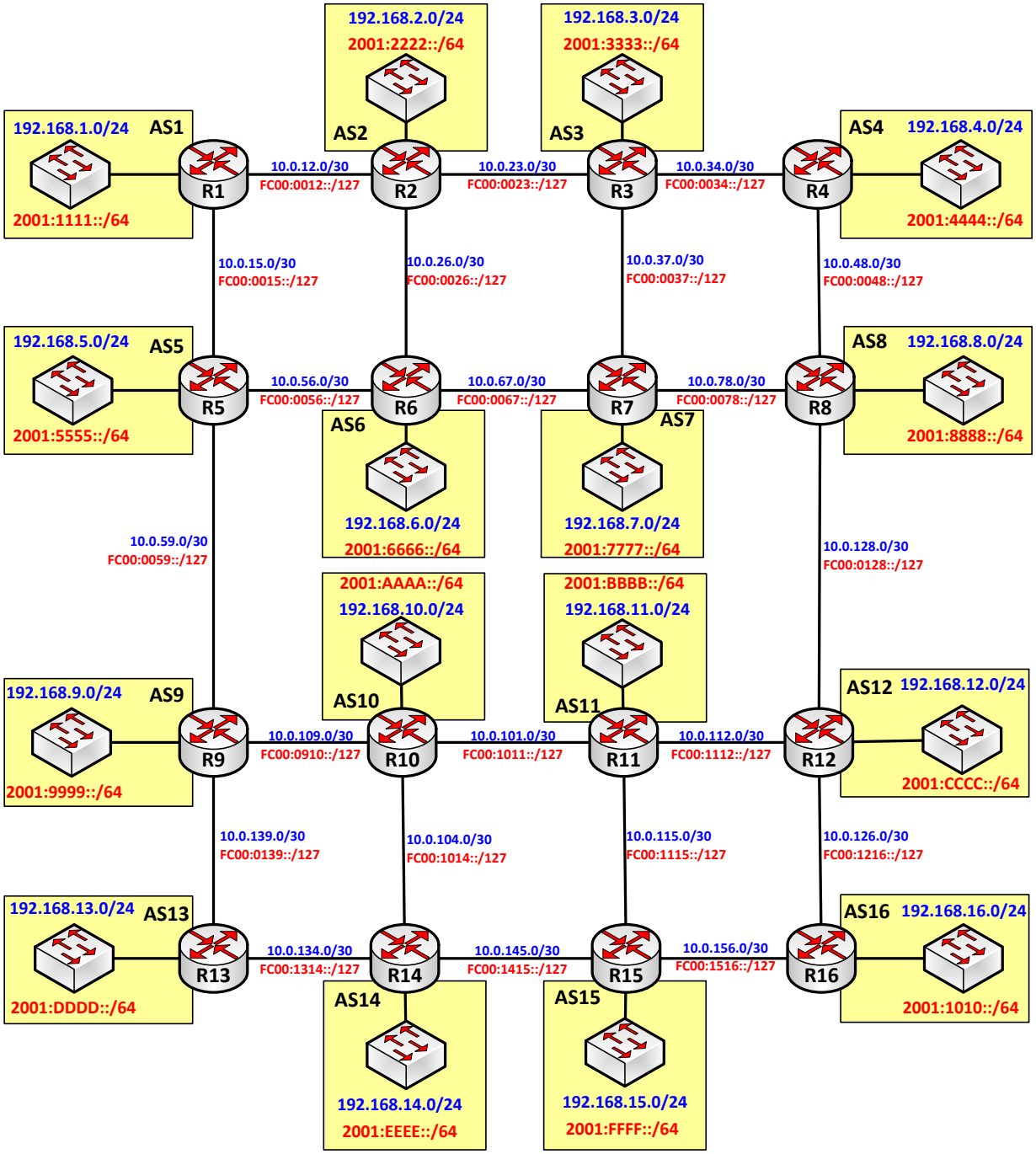


Figure 1.48

Chapter References

[1] William Slater III. Internet History and Growth. http://www.isoc.org/internet/history/

[2] UCLA Dr. Leonard Kleinrock displays Internet first router. (2009). https://www.youtube.com/watch?v=yU9oMOcRsuE

[3] Vinton Cerf, Yogen Dalal, Carl Sunshine. (1974). RFC 675 Specification of Internet. Transmission Control Program. http://www.faqs.org/rfcs/rfc675.html

[4] Radia Perlman. (1999). Interconnections: Bridges, Routers, Switches, and Internetworking Protocols. Addison-Wesley.

[5] The 3-napkins protocol. Photocopy of original autographed by Yakov Rekhter. 12-7-2007

[6] Yakov Rekhter picture. MPLS & Ethernet 2013 World Convention. Paris. https://www.youtube.com/watch?v=DcBtot5u_Dk

[7] Toni Li. Google Scholars. https://scholars.google.ca/citations?hl=en&user=BFNkOrcAAAAJ

[8] Y. Rekhter. T. Li. (1995). A Border Gateway Protocol 4 (BGP-4). RFC 1771.

[9] John T. Moy. OSPF. (1998). Anatomy of a Routing Protocol. Addison-Wesley.

[10] The CIDR Report. (2016). http://www.cidr-report.org

[11] Cogent Communications. http://www.cogentco.com/en/network

[12] Level 3 Communications. http://maps.level3.com/default/#.UgTk9ZK1GSo

[13] American Registry for Internet Numbers. https://www.arin.net/resources/request/asn.html

[14] Toronto Internet Exchange. https://www.torix.ca

[15] Ontario Research and Innovation Optical Network ORION. http://www.orion.on.ca

CHAPTER 2

BGP Basic Mechanisms

In this section, a simple network is configured, step by step, with the intention of revealing the basic mechanics of BGP. Two border routers will be talking BGP to exchange routing knowledge and this conversation will be captured and analyzed. The fundamental messages used by BGP to carry routing information are deconstructed to show their internal details. This dialog and the observation of the routing information tables illustrate BGP behaviour. The following figure shows the basic topology.

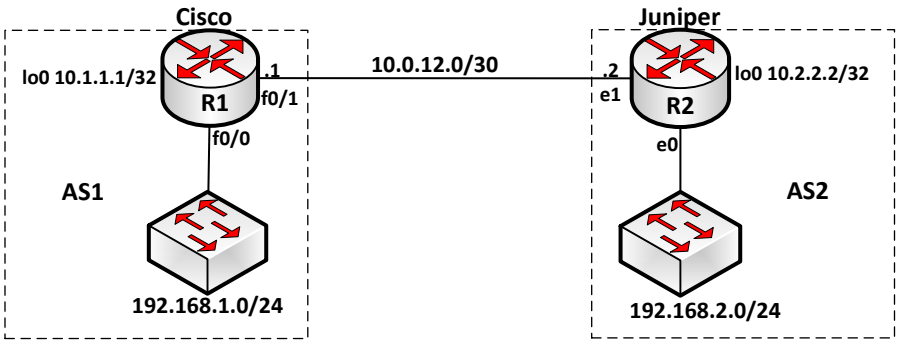


Figure 2.1

The first step is to configure the Cisco router R1's interfaces only. BGP is not configured yet.

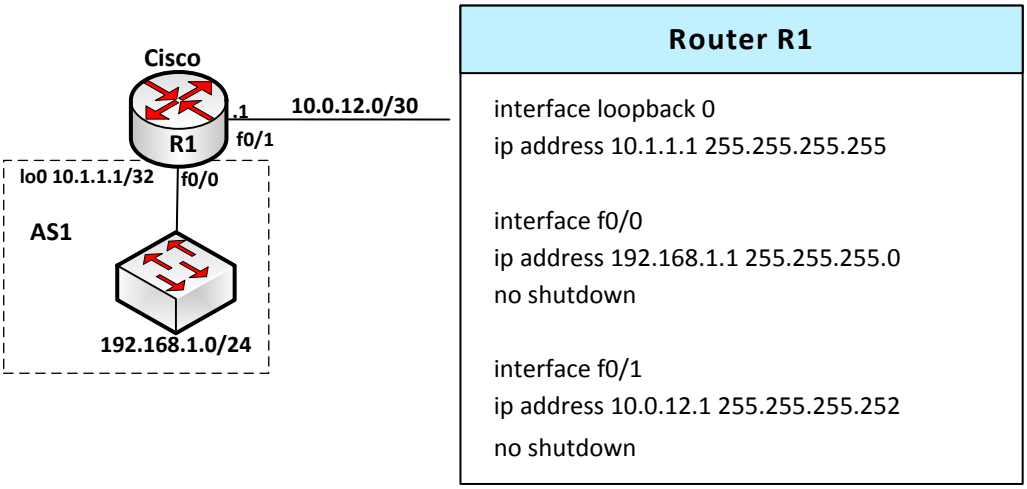


Figure 2.2

In this moment, a sniffer application (Wireshark) is started to capture the traffic on R1's f0/1 interface. Debugging is also activated in R1 to observe the behaviour of the processes related to BGP.

```
R1#debug bgp all events
BGP events debugging is on

R1#debug ip tcp transactions
TCP special event debugging is on
```

Next, BGP will be configured in R1. This is what happens as soon as BGP is activated:

```
R1(config)#router bgp 1

BGP: Initializing MDT SAFI
BGP: reset all neighbors due to BGP protocol initialization
BGP: Performing BGP general scanning
BGP (0): scanning IPv4 Unicast routing tables, IPv6 Unicast routing tables, scanning other tables.
BGP (IPv4 Unicast): computed best paths, table version went from 1 to 1
BGP (IPv6 Unicast): computed best paths, table version went from 1 to 1
```

The BGP software code in the Cisco router is undergoing an initialization process that includes finding operating resources and collecting the information located in different tables. BGP tables for both IPv4 and IPv6 are started and assigned the number one. However, the BGP database are empty since no routing knowledge have been introduced into the process as observed next:

```
R1# show ip bgp
  "nothing here"
```

The BGP process keeps checking if there is information to be collected, it tries repeatedly regulated by a timer.

```
BGP (5): ... start import cfg version = 0
BGP (5): Import timer expired. Walking from 0 to 1
BGP: notified IGPs about convergence
BGP (IPv4 Unicast): Performing BGP Next_Hop scanning for general scan
```

Now, it is the turn to incorporate routing knowledge into the BGP process even though there is no neighbor assigned yet. There are three ways by which BGP learns IP routing information:

1. From another BGP peer.
2. From the internal domain, via a manual action of the administrator.
3. From the internal domain, via redistribution from an IGP into BGP.

In a Cisco device, routing information is injected into BGP by typing:

```
R1#(config-router) # router bgp 1
R1#(config-router) # network 192.168.1.0 mask 255.255.255.0
```

This network command instructs the router to look up the prefix 192.168.1.0/24 in the IP routing table and if an exact match is found, then the IP prefix and its mask are introduced into the BGP table. BGP will only advertise the prefix if it is present in the IP routing table and that makes sense since only reachable routes should be advertised. Therefore, BGP introduces the routing information into a database which is shown next.

R1# **show ip bgp**

BGP table version is 2, local router ID is 10.1.1.1. Status codes: **s** suppressed, **d** damped, **h** history, *** valid, > best**, **i** – internal, r RIB-failure, **S** stale

Origin Codes: **i** – IGP, **e** – EGP, **?** – incomplete

Network	Next Hop	Metric	Local Pref	Weight	Path
*> 192.168.1.0	0.0.0.0	0		32,768	i

The prefix 192.168.1.0/24 has been introduced into the BGP process and it is ready to be advertised to another neighbour. The Cisco implementation of the BGP database table contains several fields that deserve commenting here:

- The prefix is marked with an asterisk (*) meaning that it is a **valid route**.
- It is also marked with the > symbol, meaning that this is the **best selection** of the route. A route can appear several times in a BGP table, but only the best, the one that appears in the IP routing table, is advertised to another BGP neighbour.
- The **Next Hop** is the address pointing to the router that has advertised the prefix, but since 192.168.1.0/24 is directly connected to this router, then the Next Hop is set to 0.0.0.0 which means itself (R1).
- The **metric** of the prefix is zero since it is directly attached to this router R1. Had the prefix being learned from another mean, then the metric could have another value.
- The **Weight** is a Cisco knob used in the implementation of policies for exit paths. The default value for directly connected routes is 32,768.
- The final field in the BGP table is the **Path** which is internal to this local autonomous system. This purpose of this field is to show the numbers of the autonomous systems leading to the destination route.

Once that the structure of the BGP database has been explained, it is time to continue with the configuration of the router. The next step is to advertise the routing information to the neighbour Juniper router R2:

```
R1(config-router)# neighbor 10.0.12.2 remote-as 2
```

Immediately, the router shows the following:

```
TCB681BFA38 created
TCB681BFA38 bound to UNKNOWN.179
TCP: starting TCP Listener process
Reserved port 179 in Transport Port Agent for TCP IP type 0
BGP: 10.0.12.2 went from Idle to Idle
```

R1 has created the corresponding TCP and BGP processes by assigning resources to this BGP session. For example, the TCP port 179 for listening BGP messages has been started. However, the BGP process for neighbour 10.0.12.2 has gone from an IDLE state back to an IDLE state. The following command corroborates the information observed from the debugging:

```
R1# show ip bgp
BGP neighbor is 10.0.12.2, remote AS 2, external link, BGP version 4,
remote router ID 0.0.0.0 BGP state = IDLE.
```

BGP starts in IDLE, it sets the resources and tables, and then attempts to continue with the connection process but in this case, the neighbour is not even configured at all, so R1 can only send ARP requests toward R2 to resolve the address 10.0.12.2 but those go unanswered.

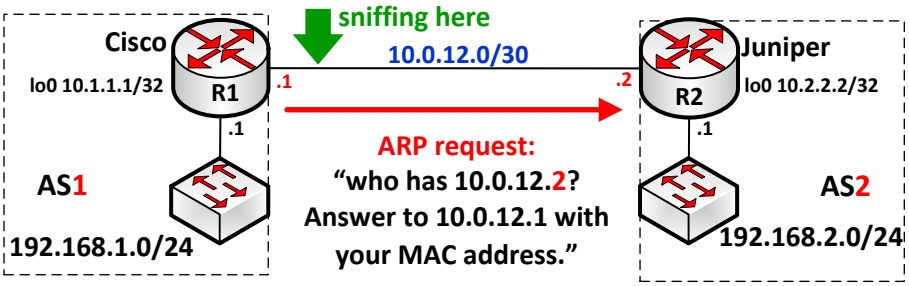


Figure 2.3

To be able to set up a TCP session, the underlying layers must be fully operational so in this case, the process stalls in IDLE until the administrator fixes the configuration on the other side (R2). Thus, the configuration of router R2 begins now with the interfaces first.

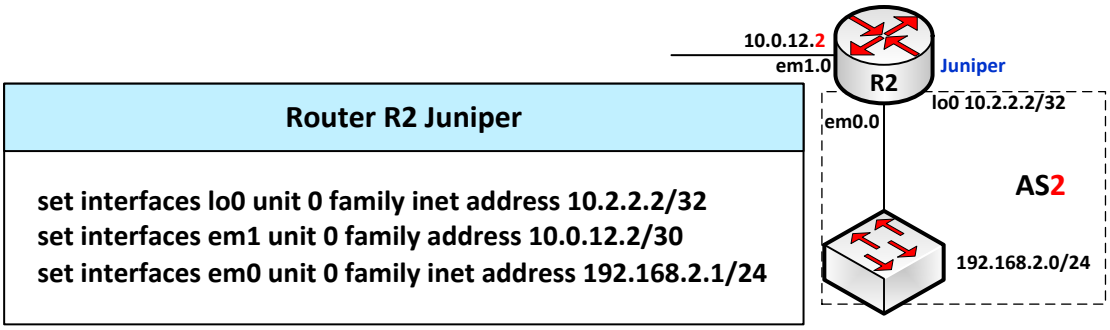


Figure 2.4

Since R2 has basic IP connectivity, it can reply to the ARP requests. Thus, R1 has now a route to talk to its neighbour so it tries to open a TCP session to the BGP port in the destination. R1 waits 5 minutes for neighbour R2 to establish a BGP connection as the following debugging shows:

```
R1# BGP (IPv4 Unicast): will wait 300s for the first peer to establish
```

Obviously, there is no answer since BGP has not been configured yet in R2. In the meantime:

```
R1#show ip BGP neighbor 10.0.12.2

BGP neighbor is 10.0.12.2, remote AS is 2, and the link is external
BGP version is 4, remote router ID is 0.0.0.0
BGP state is ACTIVE
Hold Time is 180 seconds and Keepalive interval is 60 seconds
```

The BGP process is in an **ACTIVE state**. That means that R1 is actively seeking out to connect with the neighbour R2 via TCP. Additionally, the command shows the statistics of BGP messages both sent and received. In this case, no BGP messages have been exchanged since the initial TCP connection is not even happening because R2 is not ready yet.

```
BGP Messages          Sent    Rcvd
Opens:                   0       0
Notifications:           0       0
Updates:                 0       0
Keepalives:              0       0
Route Refresh:           0       0
Total:                   0       0
The RIB (Routing Information Base) does have a route to 10.0.12.2
Connections established 0
Connections dropped 0
Last reset never
No ACTIVE TCP Connection
```

The final part of the command output is revealing for troubleshooting purposes. It says that there are no active TCP connections and that none has ever been established but that there is a logical path to the destination neighbour IP address 10.0.12.2. The debugging of the BGP transactions, with comments, helps to understand the current situation:

Sending **TCP SYN**, sequence 2479971053, ack 0. (R1 tries to start a TCP session.)

Connection to 10.0.12.2:179 advertising MSS=1460. (Destination is R2's address, TCP port 179)

State was **CLOSED**. BGP: 10.0.12.2 open failed. (But it fails!)

Connection refused by remote host, **open active** delayed. (R2 did not accept the connection.)

Furthermore, the sniffer shows what is happening at the conversation level. Router R1 sends a TCP SYN to start a session. The TCP protocol in router R2 receives the TCP SYN and forward the call to port 179 (BGP), but since BGP has not been configured in R2 the connection is refused. Consequently, R2 answers back with a TCP reset message (RST). R1 sets a timer and when it expires, it tries again to open a TCP session with a SYN, then R2 answers in the same manner because nothing has changed. The two routers are stuck in this SYN/RST situation and only the intervention of the network administrator can solve it.

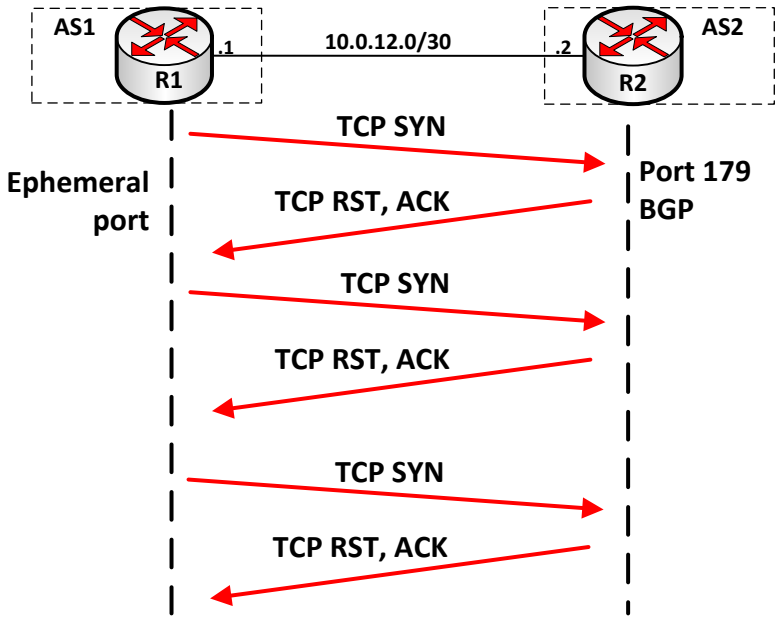
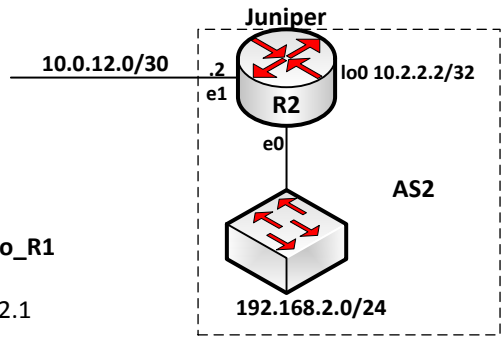


Figure 2.5

It is time to complete the configuration in the side of router R2 with the BGP settings:

Router R2 Juniper



set routing-options router-id **10.2.2.2**
set routing-options autonomous-system **2**

set protocols bgp group **peer_with_R1** type external
set protocols bgp group **peer_with_R1** export **Export_to_R1**
set protocols bgp group **peer_with_R1** peer-as 1
set protocols bgp group **peer_with_R1** neighbor 10.0.12.1

set policy-options policy-statement **Export_to_R1** term 1 from route-filter 192.168.2.0/24 exact
set policy-options policy-statement **Export_to_R1** term 1 then accept

Figure 2.6

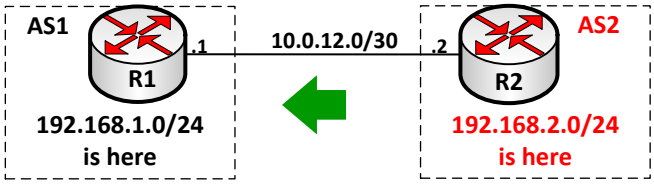
As soon as the configuration is completed in R2, the state of the BGP relation changes to:

R1#BGP-5-**ADJCHANGE**: neighbor 10.0.12.2 is **UP.**

R1 and R2 are adjacent now and soon a session is ESTABLISHED.

R1# show ip bgp neighbors
BGP neighbor is 10.0.12.2, remote AS 2, external link
BGP version 4, remote router ID 10.2.2.2
BGP State = ESTABLISHED
Hold time is 90 seconds, keepalive interval is 30 seconds
Neighbor capabilities:
    Route refresh: advertised and received
    New ASN capability: advertised Address
    Family IPv4 Unicast: advertised and received

Routing information should have been exchanged following the establishment of the session. To verify this status, the IP routing tables and BGP tables are compared in the next figures.



Router R1 Cisco

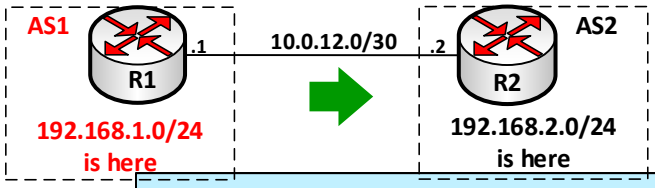
R1# **show ip route**
C 10.0.12.0/30 is directly connected, f0/1
C 10.1.1.1/32 is directly connected, lo0
C 192.168.1.0/24 is directly connected, f0/0
B 192.168.2.0/24 [20/0] via 10.0.12.2

R1# **show ip bgp**

Network	Next hop	Metric	Weight	Path
*> 192.168.1.0	0.0.0.0	0	32768	i
***> 192.168.2.0**	**10.0.12.2**		**0**	**2 i**

Figure 2.7

Next, the Juniper's table of information is shown:



Router R2 Juniper

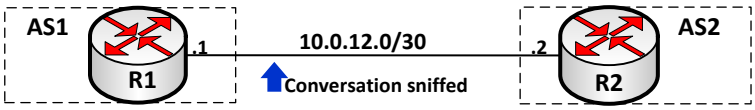
J2# **show route terse**

A	Destination	P	Prf	Metric 1	Metric 2	Next hop	AS Path
*	10.2.2.2/32	D	0			> lo.0	
*	10.0.12.0/30	D	0			> em1.0	
*****	**192.168.1.0/24**	**B**	**170**	**100**	**0**	**> 10.0.12.1**	**1 i**
*	192.168.2.0/24	D	0			> em0.0	

Figure 2.8

The networks 192.168.1.0/24 and 192.168.2.0/24 have been mutually advertised and the routers have installed the corresponding prefixes in their IP routing tables. The BGP routing exchange has been completed. More details of the BGP messages can be analyzed since the conversation was sniffed from the beginning.

The following diagram shows the sniffed BGP traffic between R1 and R2. The initial state of the conversation is a RST Reset message sent by R2 but right at that point, router R2 configuration was completed and the conversation got started.



Source IP	Destination IP	Protocol	Port number	Port number	Message
10.0.12.1 (R1)	10.0.12.2 (R2)	TCP	from 49556 (ephemeral)	to 179 (BGP)	SYN
10.0.12.2 (R2)	10.0.12.1 (R1)	TCP	from 179 (BGP)	to 49556 (ephemeral)	RST
10.0.12.2 (R2)	10.0.12.1 (R1)	TCP	from 63235 (ephemeral)	to 179 (BGP)	SYN
10.0.12.1 (R1)	10.0.12.2 (R2)	TCP	from 179 (BGP)	to 63235 (ephemeral)	SYN, ACK
10.0.12.2 (R2)	10.0.12.1 (R1)	TCP	from 63235 (ephemeral)	to 179 (BGP)	ACK
10.0.12.2 (R2)	10.0.12.1 (R1)	**BGP**	from 63235 (ephemeral)	to 179 (BGP)	**OPEN**
10.0.12.1 (R1)	10.0.12.2 (R2)	**BGP**	from 179 (BGP)	to 63235 (ephemeral)	**OPEN**

Not ready

TCP 3-way handshake

BGP begins

Figure 2.9

When two routers attempt to open a TCP session for BGP at the same time, the router with the lower IP interface address should yield its attempt to open to the router with the higher interface IP address. So router R2 starts the conversation by sending a TCP **SYN**chronize message to R1 to open a TCP session one way. Router R1 accepts the session with an **ACK**nowledge message and it asks to open a TCP session, going to the other way, with a SYN. the TCP three way handshake is completed when R2 accepts with an ACK.

R2 continues with a BGP OPEN message and router R1 follows immediately with its own OPEN.

Source IP	Destination IP	Port number	Port number	Message
10.0.12.2 (R2)	10.0.12.1 (R1)	from 63235 (ephemeral)	to 179 (BGP)	**BGP OPEN**
10.0.12.1 (R1)	10.0.12.2 (R2)	from 179 (BGP)	to 63235 (ephemeral)	**BGP OPEN**

In these OPEN messages, the routers are introducing themselves and also they are showing their timers and capabilities. They proceed to examine the contents of the OPENs to determine if they are compatible to continue the conversation and to set up timers if there is a need to do so. Also, they start preparing the exchange of routing information. In the meantime, they exchange BGP KEEPALIVE messages, sort of saying: "hey, I am still here, I am thinking, give me a second."

Source IP	Destination IP	Port number	Port number	Message
10.0.12.2 (R2)	10.0.12.1 (R1)	from 63235 (ephemeral)	to 179	**BGP KEEPALIVE**
10.0.12.1 (R1)	10.0.12.2 (R2)	from 179 (BGP)	to 63235	**BGP KEEPALIVE**

In this particular conversation, R1 reacts faster by sending a BGP UPDATE message. Router R2 replies with a TCP ACK to confirm that the message was correctly received. This is a case of BGP using the reliable capability of the transport services offered by TCP.

Source IP	Destination IP	Port number	Port number	Message
10.0.12.1 (R1)	10.0.12.2 (R2)	from 63235 (ephemeral)	to 179 (BGP)	**BGP UPDATE**
10.0.12.2 (R2)	10.0.12.1 (R1)	from 179 (BGP)	to 63235	**TCP ACK**
10.0.12.2 (R2)	10.0.12.1 (R1)	from 179 (BGP)	to 63235	**BGP UPDATE**
10.0.12.1 (R1)	10.0.12.2 (R2)	from 63235 (ephemeral)	to 179 (BGP)	**TCP ACK**

In the current topology the routing information fits into the payload of a TCP segment. That is why, there was only two UPDATES observed. But for a full Internet routing table of more than half a million prefixes, many UPDATES would be required.

From this point on and unless something changes in the network conditions, the messages exchanged are only periodic KEEPALIVES and TCP ACKs.

Source IP	Destination IP	Port number	Port number	Message
10.0.12.2 (R2)	10.0.12.1 (R1)	from 179 (BGP)	to 63235	**BGP KEEPALIVE**
10.0.12.1 (R1)	10.0.12.2 (R2)	from 63235 (ephemeral)	to 179 (BGP)	**TCP ACK**
10.0.12.1 (R1)	10.0.12.2 (R2)	from 63235 (ephemeral)	to 179 (BGP)	**BGP KEEPALIVE**
10.0.12.2 (R2)	10.0.12.1 (R1)	from 179 (BGP)	to 63235	**TCP ACK**

So far, the BGP exchange of messages has been described in a general conversation form; it is time to get deeper into the details of the Border Gateway Protocol and its messages.

BGP Encapsulation and Protocol Header

BGP is a client of the service offered by TCP which is basically reliable, connection oriented transport of data segments. TCP talks to BGP in assigned port 179 $_{decimal}$ (B3$_{hex}$). TCP is always encapsulated by IP (either version 4 or 6) which can be encapsulated in the payload of pretty much any layer 2 protocol. This is the general encapsulation of BGP from top to bottom layer.

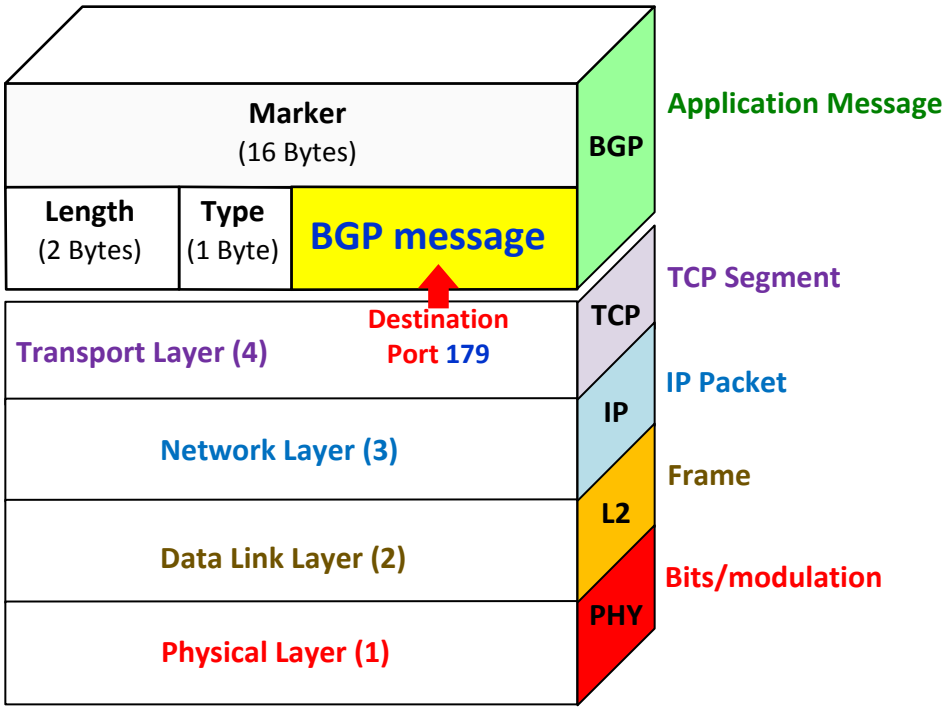


Figure 2.10

The BGP basic header has a very simple format with only three fields plus the payload with the message. The first 16 bytes is the **Marker** field set to binary ones by default. The second field is the **Length** which indicates the size of the BGP message up to a maximum of (2^{16}) 65,536 bytes long. Finally, the **TYPE** field contains a code number from 0 up to 255 but only 5 codes have been assigned for the five BGP messages that exist: **OPEN, UPDATE, NOTIFICATION, KEEPALIVE, and ROUTE REFRESH**. All of these message begin with the common BGP header.

The following diagram shows the BGP basic header. The payload of the header contains either one of the five BGP messages.

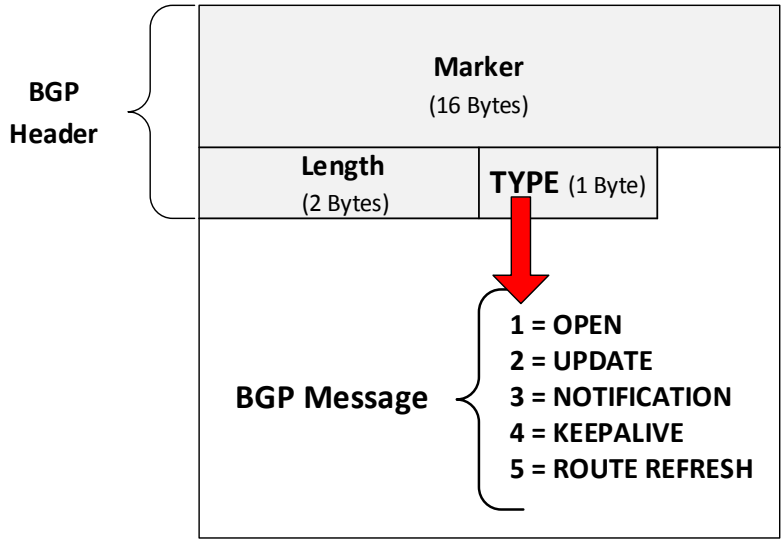


Figure 2.11

BGP OPEN Message

The first message to be examined is the **OPEN** message or BGP message **Type** number one. OPEN, as the name indicates, acts as a presentation card of router identity and what it can do. The first 19 bytes (highlighted in gray in the diagram) belong to the general BGP header. The OPEN message actually starts with the supported version of BGP which is always four (4). The next field contains the number of the Autonomous System associated to the router.

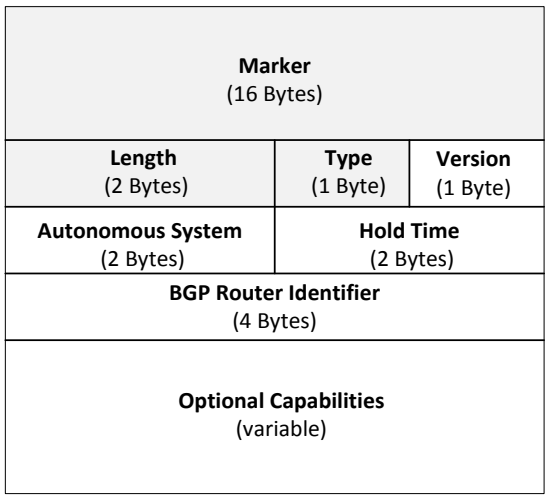


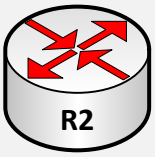
Figure 2.12

The **Hold Time** is the period of time that a BGP router waits before concluding that a neighbour is down. Peers are expected to exchange BGP messages within the Hold Time. Even when routers have nothing to say to each other, they send KEEPALIVE at a rate of three (3) messages per Hold Time. For example, if the Hold Time is 180 seconds, there is the expectation of seeing KEEPALIVES every 60 seconds. Other BGP messages such as UPDATE and NOTIFICATION can count as KEEPALIVES.

The BGP **router identifier** is an IP address. If the router is left alone, it will pick the identification from one of its directly connected interfaces. But the best practice is that the router identifier is tied to a loopback interface with an IP address assigned from a well-designed management plan. Both Cisco and Juniper have commands to manually assign the **BGP router-id**.

The final fields of the OPEN message describe **Optional Capabilities** that the router is able to implement.

BGP Header
Marker 16 Bytes
Length 53 Bytes
BGP Message TYPE number 1 = OPEN



BGP version is 4
My Autonomous System number is 2
My Hold Time is 90 seconds
My BGP router identification number is 10.2.2.2
These are my Optional parameters:

- **Capability Advertisement: I am Multiprotocol, I understand IPv4**
- **Capability Advertisement: I am Multiprotocol, I understand IPv6**
- **Capability Advertisement: I am Multiprotocol, I understand VPNs**
- **Capability Advertisement: I understand 4 Bytes AS numbers**

Figure 2.13

The optional capabilities [3] fields has played a very important role in extending the functionality of BGP over the years, as new needs have arisen, new capabilities have been added. For example, by default, a router understands IPv4 Unicast and 2 bytes autonomous system numbers. However, understanding IPv6 and 4 bytes autonomous systems might be options that needs to be activated depending on the implementation by the vendor.

In summary, by sending an OPEN message, the Juniper router R2 is basically saying "I speak BGP version 4, I am in AS2, my Hold Time is 90 seconds, my identification number is 10.2.2.2 and I also have optional capabilities which I am describing here."

Similarly, Cisco R1 is sending an OPEN saying "my name is 10.1.1.1, I belong to Autonomous System number 1, I speak BGP version 4, and if I do not hear from you in 180 seconds then I assume that you went down and I also have all these extra capabilities." Notice that Juniper R2 proposes a Hold Time of 90 seconds while Cisco R1 default Hold Time is 180 seconds. The values that they propose are their maximum acceptable times, so Router R1 can accept the 90 seconds value. The determination of timer compatibility is part of the OPEN state in the BGP connection establishment procedure.

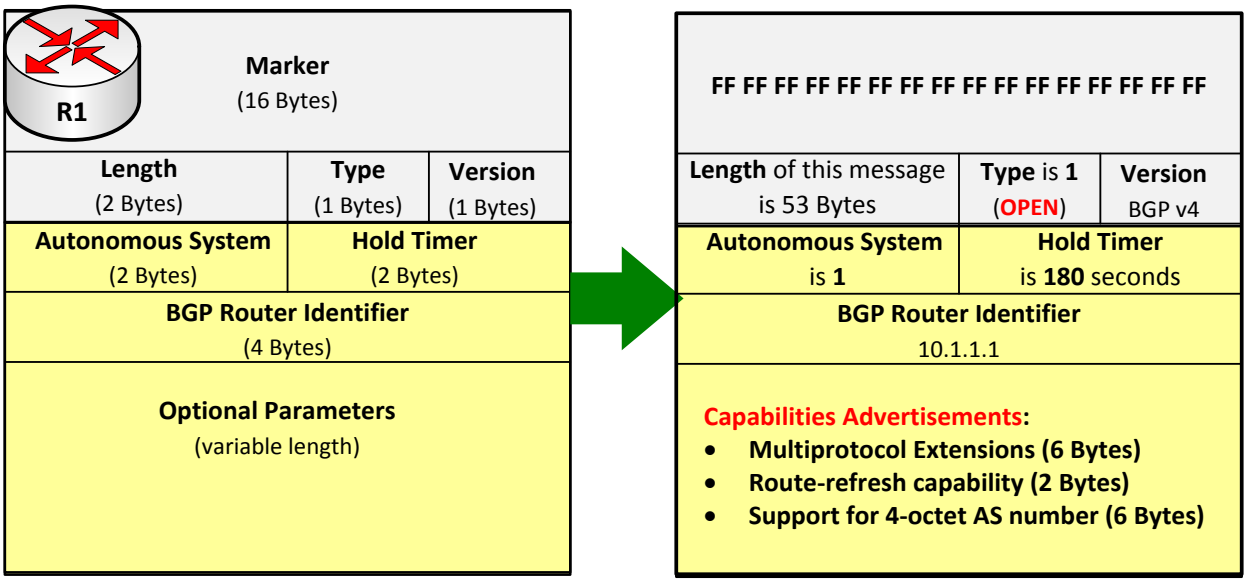


Figure 2.14

Thus R1 describes the most basic parameters to carry on this BGP conversation, but it also describes additional things that it can do. For example, R1 understand BGP Multiprotocol, Route Refresh, and extended size Autonomous System numbers. A router with older code may not understand some of the extra capabilities like the extended 4 byte AS number. If that happened, they would discover early in the conversation that there are some issues. If the disagreements were serious enough, BGP NOTIFICATION messages would be exchanged and the conversation would be terminated with TCP FIN messages being exchanged.

A clear understanding of the OPEN message and in general the OPEN state in the BGP process is fundamental for the network engineer. Many issues that occur during the session establishment process can be detected by observing the OPEN messages.

BGP UPDATE Message

The UPDATE message carries the routing information after a BGP connection has been established. The UPDATE message has two uses either to withdraw routes that are no longer valid or to advertise routes that are reachable. However, the same UPDATE message cannot be used to withdraw and to advertise simultaneously. These are the fields of the UPDATE message:

<table>
<tr><td colspan="3">Marker
(16 Bytes)</td></tr>
<tr><td>Length
(2 Bytes)</td><td>Type = 2
(1 Byte)</td><td></td></tr>
<tr><td>Unfeasible Routes Length</td><td colspan="2">Total Path Attribute Length</td></tr>
<tr><td colspan="3">Path Attributes</td></tr>
<tr><td colspan="3">Network Layer Reachability information NLRI</td></tr>
</table>

Figure 2.15

The first field is the **Unfeasible Routes Length** which is used to notify how many IP prefixes are being withdrawn in the current message. For example, if four (4) IP_{v4} networks became unreachable, the UR Length field would point to 16 bytes. Then the Network Layer Reachability Information List NLRI would list the 4 IP addresses each with 4 bytes.

When an UPDATE message is used to advertise valid routes then the same message cannot be used to withdraw any route so, in this case, the Unfeasible Routes Length is set to zero.

The **Total Path Attribute Length** indicates the number of bytes occupied by the BGP attributes which are described in the Path Attributes field. This field has a variable size because it depends on what attributes are attached to the route advertisement.

Finally, the **Network Layer Reachability Information** field (**NLRI**) carries the IP route prefix with its corresponding mask. One UPDATE can carry several prefixes, but only if they all have the same attributes. Otherwise, it is one UPDATE per prefix.

Also, to make data transfer more efficient, one TCP segment can encapsulate more than one UPDATE message. In fact, as many UPDATES as they fit in the TCP maximum segment size (mss), are piggybacked in the TCP payload.

The following snippet shows R1 sending an UPDATE message with the network prefix 192.168.1.0/24 in the NLRI field. Four BGP attributes accompany this IP route; they are ORIGIN, AS_PATH, NEXT_HOP, and MULTI_EXIT_DISCRIMINATOR. These are basic attributes, but other attributes can show up in the UPDATE depending on the particular situation.

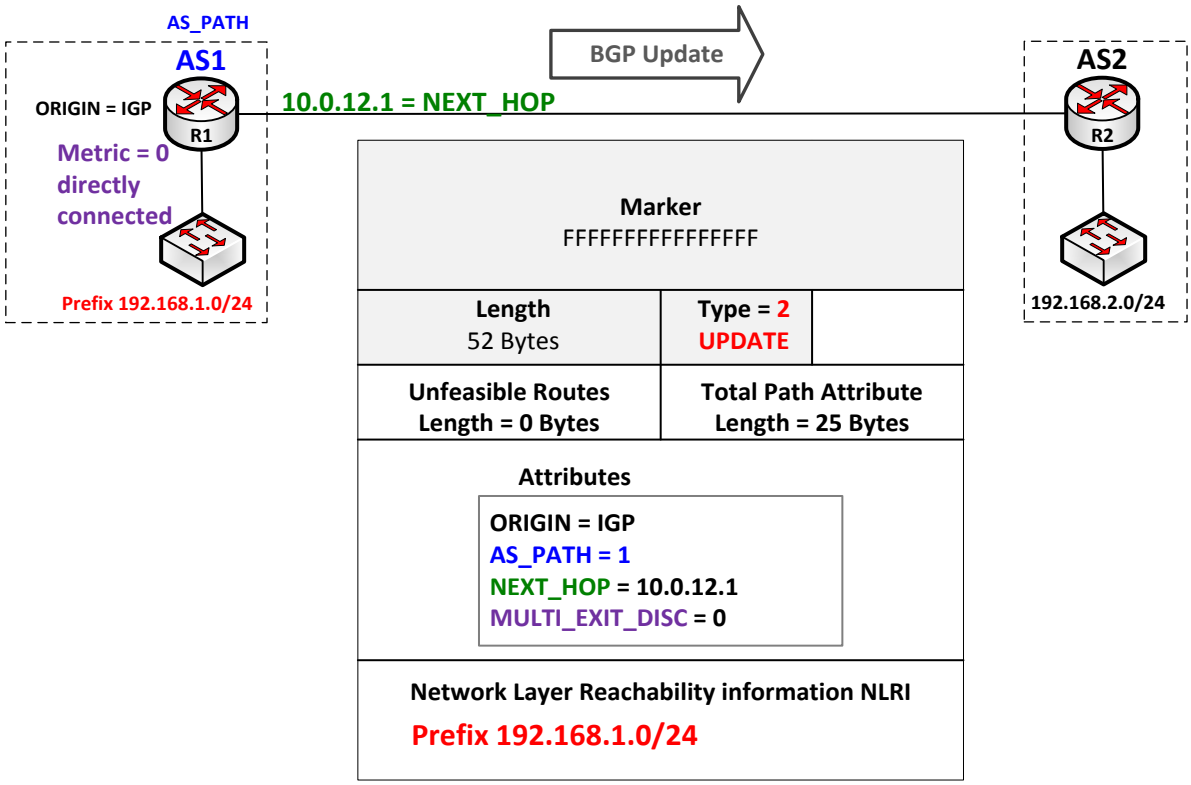


Figure 2.16

Several valid routes can be lump-summed in the same UPDATE message, but only if all the attributes are common to all of them. The following figure illustrates this point. The prefix 192.168.1.0/24, labelled A for simplicity, located inside AS1 does not have the same attributes than prefixes B and C located inside AS2 mostly because prefix A has a different AS-PATH value. Router R2 sends two UPDATE messaged to R3. One UPDATE carries the prefix 192.168.1.0/24 with an AS-PATH of AS1 and AS2. The other UPDATE carries the prefixes B and C that have common attribute values.

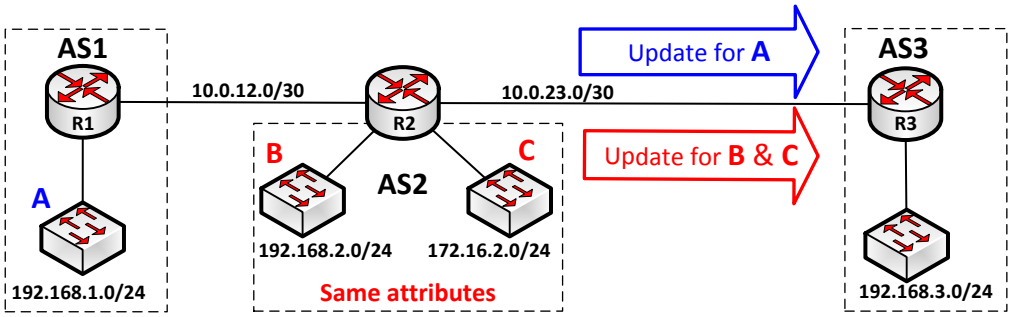


Figure 2.17

The UPDATE message has another use which is to notify when networks become unreachable. In the following example, the interface f0/0 has been shutdown to simulate a failure.

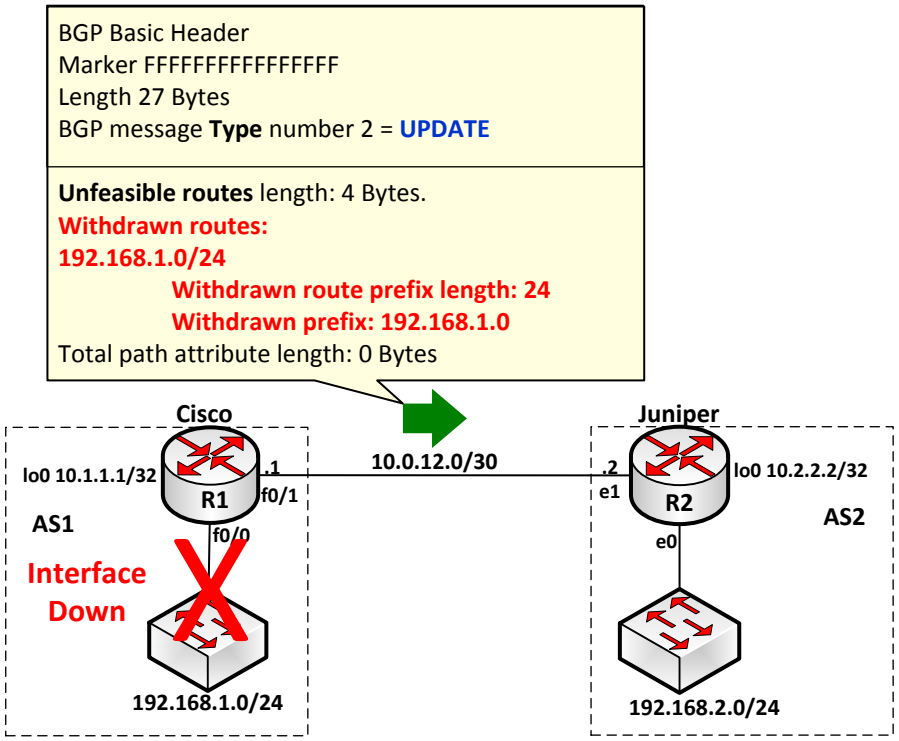


Figure 2.18

Router R1 detects that the route has to be removed from the routing table and then it proceeds to UPDATE its neighbour R2. The Unfeasible Routes Length is set to 4 bytes which is the size of one IPv4 address. The route 192.168.1.0/24 is described in the **Withdrawn Routes** fields. The attributes are not described at all. The attributes have no use when a route becomes unreachable so why bother to include them in the withdrawal message.

One UPDATE message can carry many routes that are being withdrawn as the actual limitation is the payload size of the combined TCP/IP/L2 encapsulation. A carrier level router that loses a main Internet connection would be withdrawing hundreds of thousands of routes and it will take many BGP UPDATES to accomplish that. (The Internet's routing table had 582,160 IPv4 routes on May-12-2016 when this was written).

BGP NOTIFICATION Message

NOTIFICATION is BGP message type number 3. This message is used to communicate problems that affect the BGP session such as header problems, wrong AS numbers, version not supported, failures in the authentication, and capabilities not supported. When there is a serious lack of compatibility or configuration problems, routers refuse to peer, then they notify the reason to each other. The NOTIFICATION message has two fields, the **Error Code** and the **Sub-code**. Every error case has been categorized and given a code number and each error code has a sub-code field that adds details to the notification.

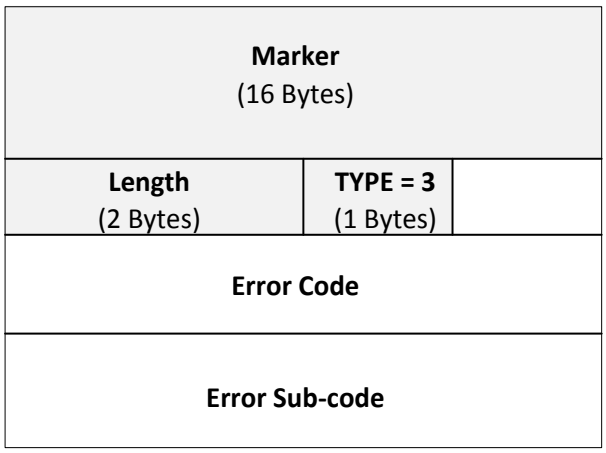


Figure 2.19

The following table lists the standard error codes as per RFC-4271. Each of these Error Codes has its corresponding table of sub-codes.

Error Code Number	Meaning
0	Reserved
1	Error in the BGP header
2	Error in the OPEN message
3	Error in the UPDATE message
4	The Hold Time has expired
5	Finite State Machine error
6	Cease
7	ROUTE-REFRESH message error
8-255	Unassigned

Table 2.

For example, if router R2 had not sent any BGP messages during the Hold Time. When the timer expires, R1 sends a NOTIFICATION message and then proceeds to close the session.

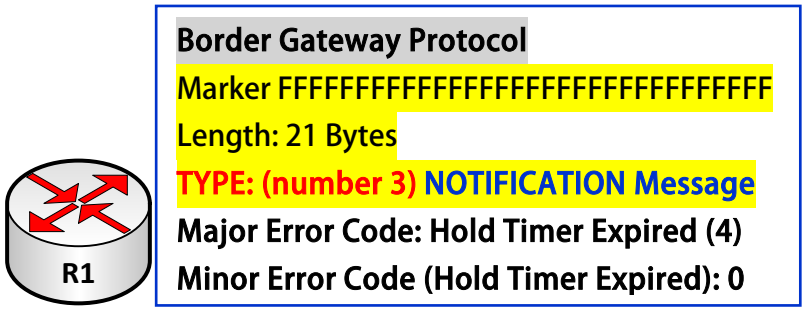
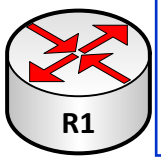


Figure 2.20

In another example, R1 detects a major configuration problem for which there is no specific error code. R1 sends a NOTIFICATION message saying that the intended connection has been rejected.



Border Gateway Protocol
Marker FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
Length: 21 Bytes
TYPE: (number 3) NOTIFICATION Message
Error Code: Cease (6)
Error Sub-code: Connection Rejected (5)

Figure 2.21

The following configuration has a mistake in R1's side. In effect, the router administrator has made a mistake in the configuration of the autonomous system number, it should be 1 not 11.

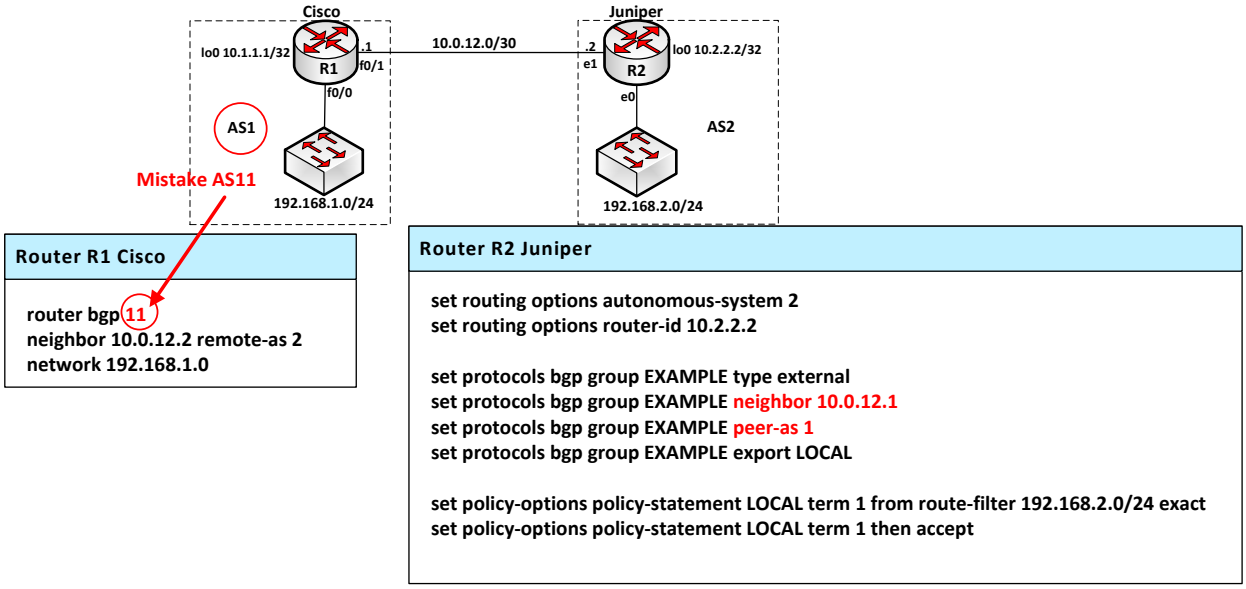


Figure 2.22

When the configuration is completed, the two routers attempt to establish a BGP session so they exchange OPEN messages that contain the AS numbers. R2 is expecting to see AS 1 in the autonomous system field of the OPEN sent by R1, but it sees AS 11 instead. Then it refuses to continue with the peering process by sending a BGP NOTIFICATION message to R1.

Border Gateway Protocol
Marker FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
Length: 21 Bytes
TYPE: (number 3) NOTIFICATION Message
Major Error Code: OPEN Message Error (2)
Minor Error Code (OPEN Message): Bad Peer AS (2)

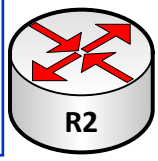


Figure 2.23

The error message 2 correspond to anything detected during the OPEN state while the sub-code gives the more specific information of what the problem is. The following table displays the logic that the router uses to create the NOTIFICATION message.

ERROR Code Number	Meaning
2	OPEN message error

OPEN Sub-code Number	OPEN Sub-code meaning
0	Unspecific
1	Unsupported version
2	Bad Peer AS
3	Bad BGP Identifier
4	Unsupported Optional Parameter
5	Deprecated
6	Unacceptable Hold Time
7	Unsupported Capability
8-255	Unassigned

Table 2.

When router R1 receives and processes the NOTIFICATION, a message is posted in the console screen to help the administrator troubleshoot the problem.

```
R1#%BGP-3-NOTIFICATION: received from neighbor 10.0.12.2 2/2 (peer in wrong AS) 0 bytes
```

R2 is notifying, in the error message, that the AS number is incorrect but the message, in the way that it is worded in IOS, is not that evident, so more troubleshooting is needed until finding which of the two AS number is incorrect. Furthermore, activating the debugging in R1 helps to understand the inner working of the BGP protocol.

```
R1#
BGP: 10.0.12.2 went from Idle to Active
BGP: 10.0.12.2 open active, local address 10.0.12.1
BGP: 10.0.12.2 read request no-op
BGP: 10.0.12.2 went from Active to OpenSent
BGP: 10.0.12.2 sending OPEN, version 4, my AS: 11, holdtime 180 seconds
BGP: 10.0.12.2 send message type 1
BGP: 10.0.12.2 remote close
BGP: 10.0.12.2 -reset the session
BGPNSF state: 10.0.12.2 went from nsf_not_active to nsf_not_active
BGP: 110.0.12.2 went from OpenSent to Idle
BGP: 10.0.12.2 closing
BGP: 110.0.12.2 went from Idle to Active
Repeat, etc.
```

The BGP process starts in **IDLE** state, and then it goes to **ACTIVE** to try to start a TCP session with R2. A TCP connection is started from source IP address 10.12.12.1 and this session is completed since from the viewpoint of TCP everything appears correct. Then R1 sends a BGP OPEN message. R2 reads in the OPEN message that the ASN is 11 when it is expecting to see AS1 instead. It prepares and sends a NOTIFICATION message to R1 explaining that it cannot become adjacent since the AS number is incorrect. Immediately, R2 sends a TCP FIN to end the TCP session.

Source	Destination	Protocol	Message	Comment
R1	R2	TCP	SYN	
R2	R1	TCP	SYN + ACK	TCP 3-way handshake
R1	R2	TCP	ACK	
R1	R2	BGP	OPEN	BGP tries to OPEN
R2	R1	BGP	OPEN + NOTIFICATION	ERROR message is sent
R2	R1	TCP	FIN + ACK	
R1	R2	TCP	ACK	TCP session closes
R1	R2	TCP	FIN + PSH + ACK	
R2	R1	TCP	ACK	
R1	R2	TCP	SYN	
R2	R1	TCP	SYN + ACK	TCP 3-way handshake
R1	R2	TCP	ACK	
R1	R2	BGP	OPEN	BGP tries to OPEN
R2	R1	BGP	OPEN + NOTIFICATION	ERROR message is sent

Figure 2.24

R1 has no choice but to close its TCP end as well, but then it insists in opening a TCP session again, the process gets into a cycle and only the intervention of the router administrator can fix the situation.

BGP KEEPALIVE Message

A KEEPALIVE is just the basic BGP header (19 bytes) with the TYPE set to number four. It does not contain anything else because it is just meant to indicate that the BGP process is alive even when there is no routing information to exchange. The reference is a counter based on the **Hold Time** value which is defined in document RFC-4271 as the time that must elapse after the last KEEPALIVE or UPDATE to consider that a BGP neighbour is down. This document also states that routers must set their KEEPALIVES at the pace of three (3) per Hold Time.

Border Gateway Protocol
Marker FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF (the marker has 16 Bytes)
Length = 19 Bytes (the length field has 2 Bytes)
TYPE: (number 4) KEEPALIVE Message (the Type field is 1 Byte)

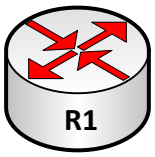


Figure 2.25

KEEPALIVES are the only BGP messages exchanged when routers are in a converged state. Each KEEPALIVE is quickly followed by a TCP ACK. The exchange of these messages and their timing is shown in the next diagram:

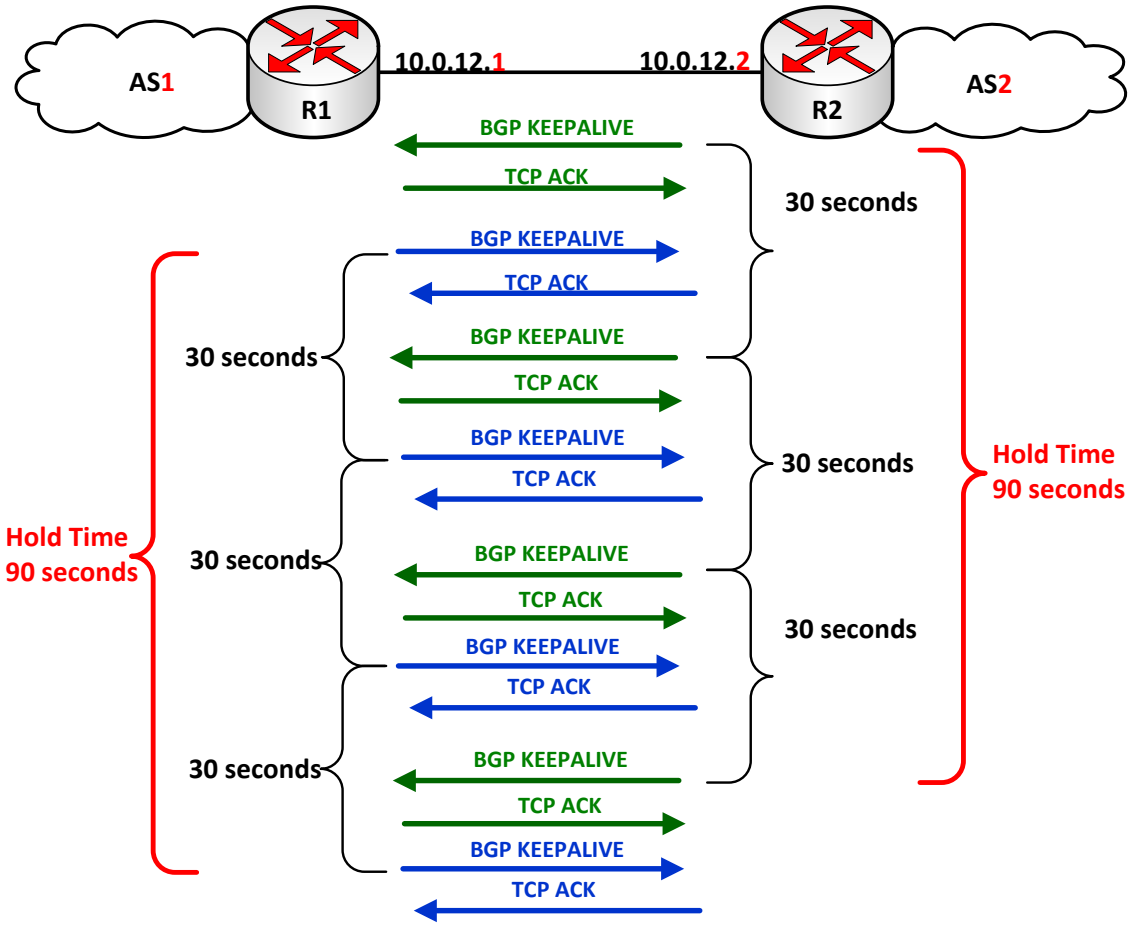


Figure 2.26

Routers advertise their Hold Times when they exchange OPEN messages. In this case, Cisco R1 proposed its maximum value of 180 seconds, but Juniper R2 said that 90 seconds was its highest value, so Cisco accepted 90 for a common value. Consequently, with a Hold Time of 90 seconds each router sends one KEEPALIVE every 30 seconds. On the other hand, a Hold Time of

180 seconds would have made the routers to send one KEEPALIVE every 60 seconds Basically, lower Hold Time causes more frequent KEEPALIVES thus detecting a neighbour's failure faster.

BGP ROUTE REFRESH Message

The last BGP message is ROUTE-REFRESH [2] or Type number 5. This is the scenario for the application of this message. Suppose that router R1 has sent a BGP UPDATE to R2 which has processed the routing information thru its local inbound policy. The result of that is installed in the BGP database and presented to IP for its installation in the IP routing table. After that UPDATE, the two routers have nothing else to say and they just exchange KEEPALIVES periodically.

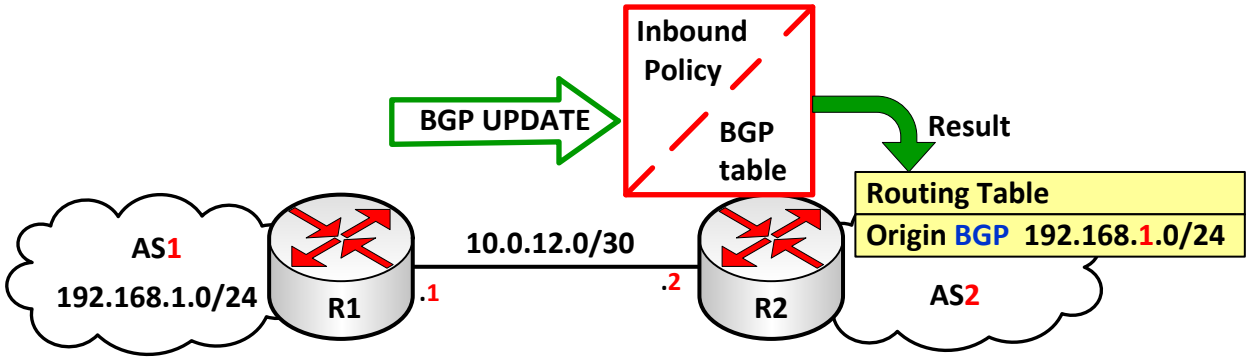


Figure 2.27

The administration in AS2 decides that a change in the inbound policy is required to filter out some prefixes, for example. The policy is modified accordingly, but the changes do not take effect until a new UPDATE is sent by router R1. The problem is that R1 has no reason to send a new UPDATE because from its viewpoint everything is working the way it should be. So the administrator in R2 needs to do something to force the new UPDATE. That is exactly the function of the ROUTE-REFRESH message which is activated with the following commands depending of the vendor hardware deployed:

Cisco# clear ip bgp 10.0.12.2 in	Juniper> clear bgp neighbor 10.0.12.1 soft-inbound

The address of the BGP neighbour must be specified in the command line because the ROUTE-REFRESH is targeted to a particular destination. (In a production environment, clearing commands must be avoided or done with extreme care.) Once the command is executed, router R2 sends a BGP ROUTE-REFRESH message.

Border Gateway Protocol – ROUTE REFRESH Message
Marker FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
Length: 23 Bytes
TYPE: (number 5) ROUTE-REFRESH Message
Address Family Identifier (AFI): IPv4 (1)
Sub-type: normal route refresh request [RFC-2918] with/without ORF [RFC-5291] (0)
Subsequent Address Family Identifier (SAFI): Unicast (1)

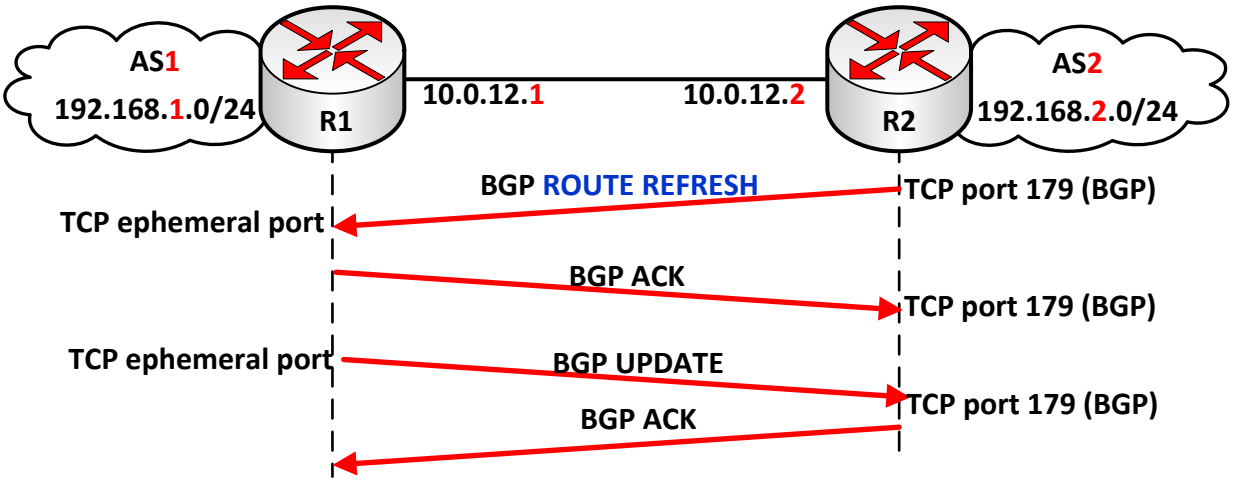


Figure 2.28

Router R1 responds with a TCP ACK and then it sends a BGP UPDATE. R2 receives the UPDATE and pass it thru the inbound policy to obtain a result that will be presented to the IP routing table eventually.

BGP Finite State Machine FSM

Protocols must be designed considering every single possible situation that might occur during the operation. Consequently, the protocol designer must follow a system to outline all the possibilities in an organized way. A Finite State Machine (FSM) is exactly that, a design technique where every possible state is taken into account. A FSM helps the designer to ponder "what if" type of questions and prepare solutions so a protocol does not break down or stall unexpectedly. Also, a FSM is helpful for troubleshooting since it allows to predict the next possible outcome that should follow the current state. BGP was designed according to the steps in a Finite State Machine and that is why the observation of the BGP connection establishment shows well defined states.

This is a **simplified version** of the **BGP Finite State Machine**. The process starts in the IDLE state as soon as the administrator configures BGP. In this state, BGP seeks out resources to operate, resets counters and timers, and instantiates events.

From IDLE, the BGP process transitions into the CONNECT stage where the protocol asks TCP to start a transport session. BGP waits for the TCP 3-way handshake but if that fails, it sets a ConnectRetry timer to attempt again. If the TCP session cannot be brought up, then BGP goes back to IDLE.

If the TCP connection is completed, BGP moves to send OPEN messages to the peer. If the peers are incompatibles, a NOTIFICATION message is prepared and sent and then BGP goes into IDLE.

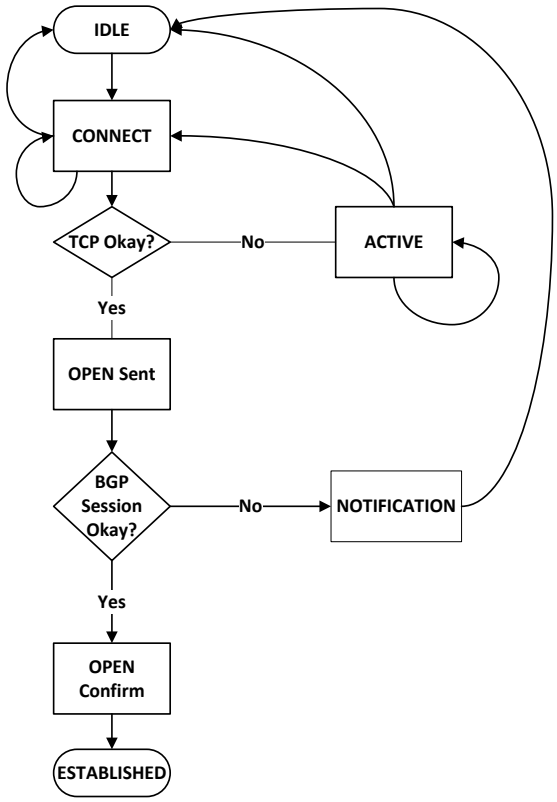


Figure 2.29

But if the BGP session is okay, then BGP move to send an acknowledgement to the neighbour confirming the BGP peering. UPDATES messages are exchanged and the session becomes ESTABLISHED. Stable, fully operating routers should be in this state. In case of any failure situation, the protocol resorts back into IDLE state.

Chapter Summary

Border Gateway Protocol follows a well-defined set of rules based on a finite state machine to initiate, follow up and terminate conversations. BGP designers defined four messages that are indispensable to carry on a BGP conversation. These messages are OPEN, UPDATE, NOTIFICATION and KEEPALIVE and they enable to say everything that there is to say in a BGP exchange. Later, based on the necessity of refreshing the inbound routing information for new policies, the message ROUTE REFRESH was added to the list. It is possible to add new messages to BGP, but the need has not arisen and the protocol has been stable in its current version (4) for a long time.

Each BGP message has a TLV (Type, Length, and Value) format with the fields needed to perform its function. The messages are encapsulated by TCP which provides the transport layer services to BGP. The designers of BGP implemented the protocol following the rigour of a finite state machine where all possible outcomes are considered so the protocol does not fail in an unrecoverable way.

BGP Mechanisms

Exercise

- Implement a simple network of only two routers in two autonomous systems:

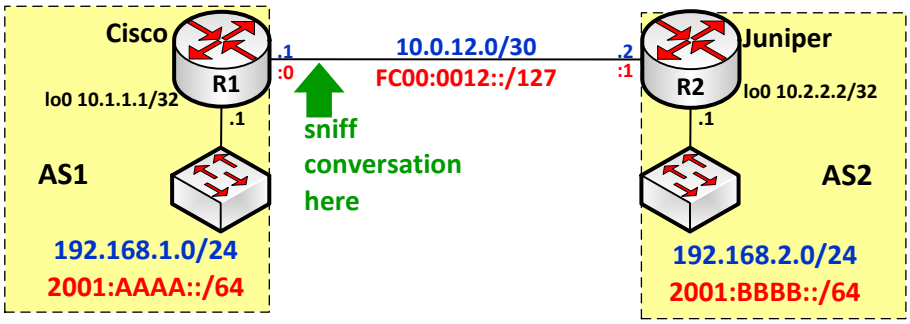


Figure 2.30

- Debug the BGP conversation between the ASBRs.
- Sniff the BGP conversation between the ASBRs.
- Follow the visible (not all states are) Finite State Machine States one of the routers
- Describe the FSM States as you see it happening.
- Explain all the BGP transactions between the ASBRs.
- Understand all the messages and their headers at the field level.
- Describe content of BGP bases and IP routing table.
- Make mistakes in the configuration such as wrong AS number and analyze the debugging of the error.

Chapter References

[1] Y. Rekhter, T. Li. S. Hares. (2006). RFC-4271. A Border Gateway Protocol 4 (BGP-4). https://tools.ietf.org/html/rfc4271

[2] E. Chen. (2000). RFC 2918. Route Refresh Capability for BGP-4. https://tools.ietf.org/html/rfc2918

[3] RFC 5492. J. Scudder. R. Chandra. (2009). Capabilities Advertisement with BGP-4. https://tools.ietf.org/html/rfc5492

[4] IANA. List of BGP Capabilities Codes. (2016). http://www.iana.org/assignments/capability-codes/capability-codes.xhtml

[5] D. Walton, D. Dutt. (2016). Internet-Draft Hostname Capability for BGP. https://tools.ietf.org/html/draft-walton-bgp-hostname-capability-02

CHAPTER 3

BGP Attributes

Conceptually, BGP considers that each network prefix is an object with attributes. The policies implemented in the routers match these attributes to either block, accept, or steer Internet traffic. This section explains the most basic BGP attributes.

The following diagram illustrates the fundamentals of BGP route advertisement. In this case, an Autonomous System contains two IP prefixes. For simplicity, they will be called A and B instead of their IP addresses. The Border Router R1 advertises prefix A with a corresponding set of attributes generically named X and Y for the effect of this explanation. Similarly, prefix B would be advertised with its own set of attributes as well.

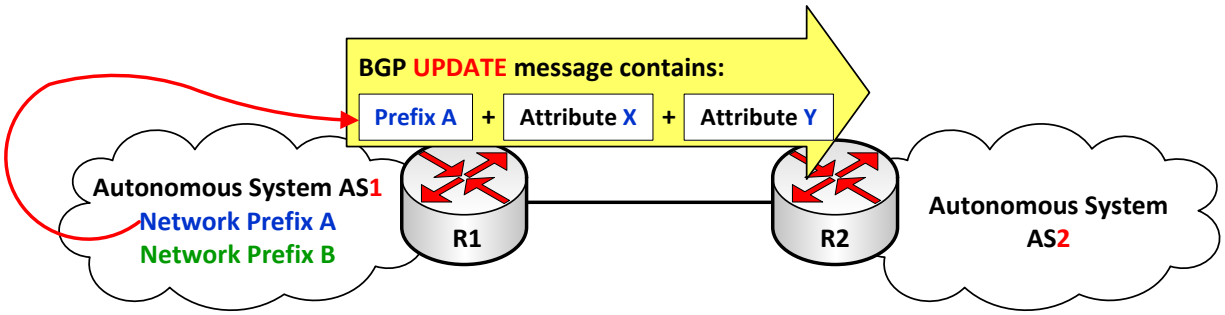


Figure 3.1

Structure of a BGP Attribute

Each attribute is designed with a standard **TLV** (Type, Length, and **Value**) format where a **Type** field indicates the attribute being described. The **Length** points out the total byte size of the value field which is variable depending of the attribute. The **Value** field is the actual attribute.

Flags	TLV (Type, Length, Value

Figure 3.2

The TLV design system enables protocol expansion because new attributes can be added simply by specifying a new type number and its properties. Several attributes have been added to the original specification of BGP and nevertheless the protocol has remained in version 4.

Additional to the TLV field, all attributes are preceded by **Flags** that hints the routers how to deal with the attribute. In more detail, an attribute has this general structure:

Flags	Type	Length	Value
One Byte	One Byte	One byte	Variable Size

Figure 3.3

Flags Field

The flag field carries information that the routers use to propagate the routing information. This field is one byte long thus providing eight possible bit-flags. However, only 4 bits have been specified. The following table describes the flags meaning:

Option flag bit	Transitiveness flag bit	Completion Flag bit	Length flag bit				Rightmost bit
0 = well known	0 = non-transitive	0 = complete	0 =regular	0	0	0	0
1 = optional	1 = transitive	1 = partial	1 = extended	0	0	0	0

Table 3.1

Examples:

- 40_{hex} is $0100\ 0000_{binary}$ means well-known, transitive, complete, and regular length.
- 80_{hex} is $1000\ 0000_{binary}$ means optional, transitive, complete, and regular length.
- 90_{hex} is $1001\ 0000_{binary}$ means optional, transitive, complete, and extended length.

Optional flag bit

- **Well known Attribute (bit position is set to zero)**
  - **Mandatory**
  - **Discretionary**

When the bit 0 of the flag field has a value of zero then the attribute is well known. Well known attributes must be understood by all BGP speaking routers so all the common BGP attributes (ORIGIN, NEXT_HOP, and AS_PATH) are of this type. Well known attributes must be recognized by all BGP implementations and the failure to understand such attributes generates BGP NOTIFICATION Error messages and the closing of the BGP session. Some of the well-known

attributes are **mandatory** because all routers must include them. The following diagram is used to illustrate this definition.

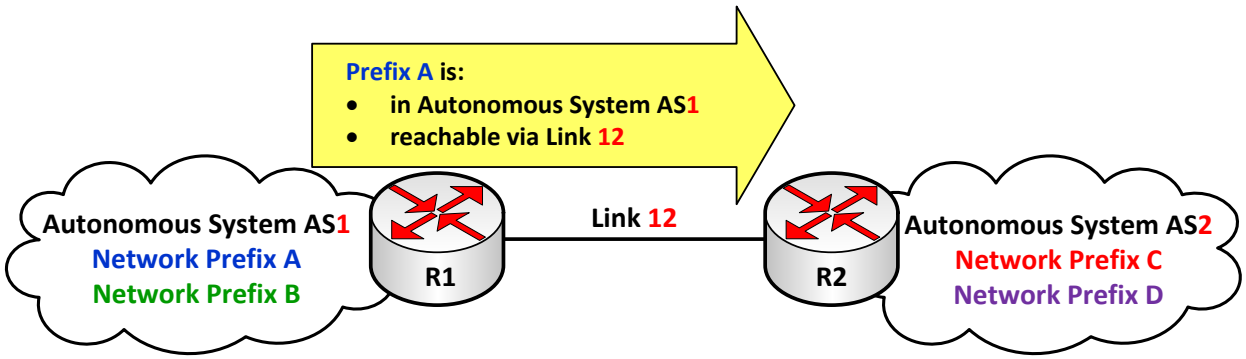


Figure 3.4

For example, Router R1 advertises prefix A with the description of where it is located. Two attributes are used for that, the number of the source autonomous system and the address of the interface that sends the UPDATE message. These two attributes are well-known and mandatory because BGP could not work without this very basic information.

Other well-known attributes are **discretionary** and they only appear in the route advertisement because of an administrative decision. For example, the network administrator of AS1 decides to supernet the prefixes A and B and to advertise the aggregation to R2 in AS2. When the aggregation is configured in R1, a new attribute can be added to the UPDATE to indicate that aggregation has been done. This attribute would be well-known but discretionary since all routers must be able to understand it, but the attribute does not have to necessarily be used.

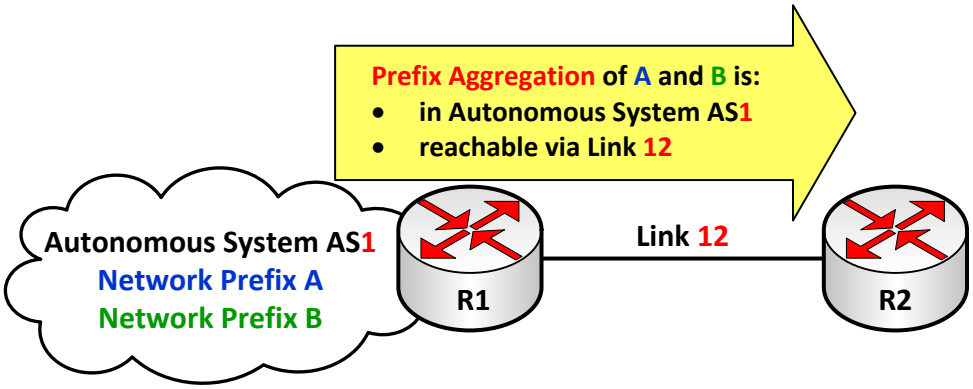


Figure 3.5

Optional flag bit

- **Optional Attribute (bit position is set to one)**

The BGP specification is open to the possibility that some attributes might be defined for specific applications or might be vendor proprietary. In those cases, understanding or implementing an action based upon the attribute would be **optional** for a router. Also, optional attributes does not have to be used all the time even if they are well known.

Flag bit transitiveness

- **Non-transitive (bit set to 0)**
- **Transitive (bit set to 1)**

Transitive means that the attribute can be passed on to another autonomous system. In the diagram below, the topology has no policies implemented to modify the behaviour of BGP.

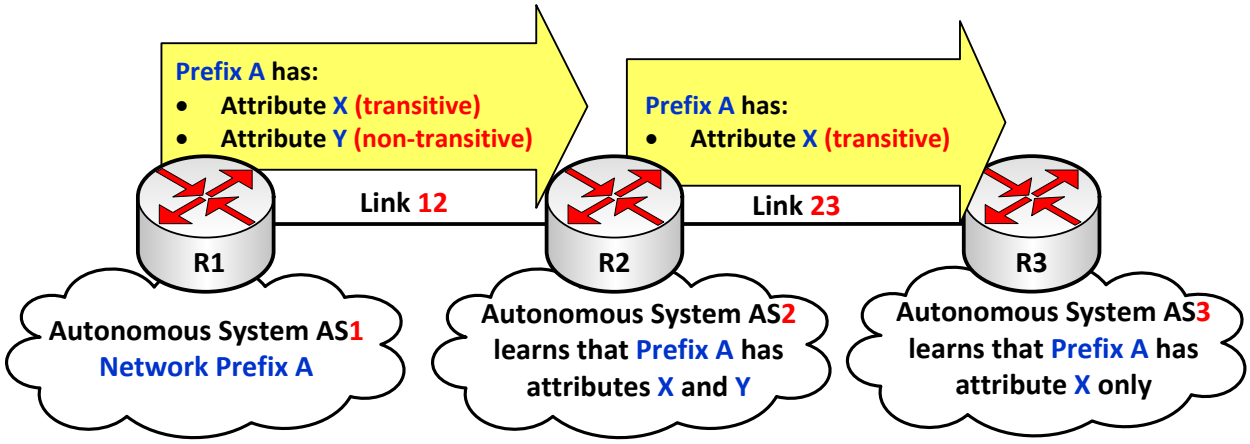


Figure 3.6

Router R1 advertises the route prefix A to router R2 which, in turn, advertises the same prefix to R3. The original UPDATE sourced by R1 carried two attributes, generically named X and Y. When R2 sends an UPDATE to R3, only the attribute X accompanies the route information because attribute X is transitive. Attribute Y was not passed along because Y is non-transitive.

Typically, non-transitive attributes have significance only to the immediate neighbour AS and that is why it makes no sense to further propagate their values. On the other hand, transitive attributes are needed to qualify the network prefix along the whole path.

Flag bit Completion

- **Complete (bit set to 0)**
- **Partial (bit set to 1)**

Router R1 advertises prefix B with two attributes W and Z, both attributes are marked transitive, so they must get to R3. Router R2 recognizes attribute W so this is a **complete** attribute because all the routers in the path understand what it is for. However, router R2 does not recognize attribute Z, but being transitive; it has to propagate the routing information to R3. Router R2 will mark the flag bit **partial** and passes it to R3 in a new UPDATE. It is sort of "I pass this along, but I do not know what it is for".

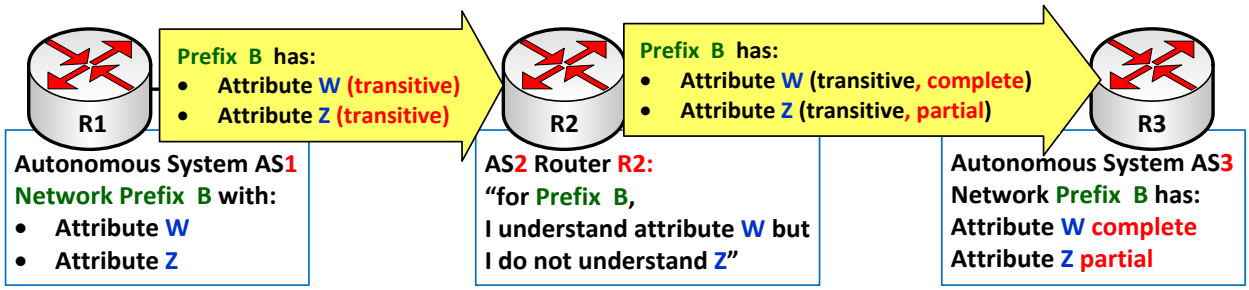


Figure 3.7

Flag bit Length

- **Regular (bit set to 0)**
- **Extended (bit set to 1)**

Attributes that were defined in the original BGP version 4 are **regular** length attributes. A flag-bit of zero means that the attribute length field has one byte to describe the size of the attribute. Attributes that have been added or modified later have the **extended** bit set to 1. For example, the 4 Bytes Autonomous System numbers that were added to counter the exhaustion of the original AS number space of only 2 Bytes. In such case, the bit 3 is set to 1 and that means that the attribute length is represented by two bytes.

BGP Attribute type field

The attribute field has 8 bits so it yields 256 possible code numbers, the majority of which are reserved for future applications. The existence of this field has contributed to the extension of the BGP protocol over the years. When a new attribute is required, a number is taken from the reserved lists and assigned to the attribute. The following list shows the most common BGP attributes:

TYPE #	Attribute Name	Main usage or meaning
1	Origin	To indicate how a route was injected into BGP
2	AS_PATH	List the autonomous systems to reach the route
3	Next_Hop	The interface of neighbour that advertised the route
4	MED	Metric send to peer to indicate path preference
5	Local-Preference	Indicate exit path preference from within the AS
6	Atomic-Aggregator	To indicate that supernetting happened
7	Aggregator	Indicates the router that did route aggregation
8	Community	To tag routes with labels. Used in policy routing.
9	Originator	Identification of router that introduces the route.
10	Cluster-List	A list of identifications of a BGP cluster of routers.
14	Multiprotocol MP	BGP is Multiprotocol. It can advertise IPv6, VPN, etc.
16	Extended Communities	Extended communities tag routes.

Table 3.2

The core of BGP is based on using the attributes in this list. For a person to implement and manage BGP routing, the thorough understanding of the attributes is a requirement. In BGP administration, some attributes appear on a daily basis, others are less frequent. In the previous list, there are three attributes that appear in every single UPDATE message and every router must understand them, they are the BGP Well-known Mandatory Attributes:

1. ORIGIN
2. AS_PATH
3. NEXT_HOP

These attributes are explained in detail in the rest of this chapter. The remaining attributes will be explained in other chapters as they are needed in particular situations.

BGP Attribute ORIGIN

ORIGIN is the first attribute that accompanies every route being advertised by BGP. As the name indicates, ORIGIN provides network administrators with an indication of how the routing knowledge came to be introduced into BGP. The document RFC-4271 specifies that this attribute is instantiated by the BGP speaker that originates the routing information and it should not be changed by any other router.

When the older BGP-version 3 was specified in RFC-1267 in 1991, the designers wanted to differentiate the information that came from within the routing domains (Interior Gateway Protocols) apart from the information that came from the protocol Exterior Gateway Protocol (EGP) which was an interdomain protocol now extinct. Additionally, the BGP designers included a third origin category for everything that did not fall within the previous classification. Consequently, ORIGIN has the following three code numbers:

Type	Type name	Meaning
0	IGP	Routes that were learned from an IGP initially
1	EGP	Routes originally advertised by EGP (deprecated)
2	INCOMPLETE	Routes that do not comply with the previous categories

Table 3.3

ORIGIN is a well-known attribute so all BGP speakers must understand it. ORIGIN is transitive so all BGP speakers must pass it along and it is complete so there is nothing to add to it. The following fields of the BGP UPDATE message describe an ORIGIN attribute.

Flags	Type	Length	Possible Values
0 = well known 1 = transitive 0 = complete 0 = regular length	Type Code 1 (Origin)	1 Byte	0 = IGP 1 = EGP 2 = incomplete

Table 3.4

The details of structure and meaning of an ORIGIN attribute are better observed inside an actual UPDATE message. The following topology is used to test the implementation of the ORIGIN Attribute in both Cisco and Juniper.

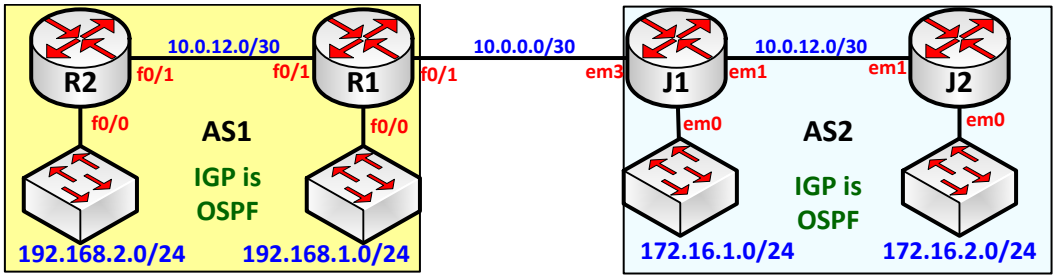


Figure 3.8

Description

- Routers R1 and R2 are Cisco 7200 routers.
- Routers JUNOS1 (abbreviated J1) and JUNOS2 (abbreviated J2) are Juniper routers.
- R1 advertises to J1 the prefixes 192.168.1.0/24 and 192.168.2.0/24 only.
- R1 introduces into BGP the prefix 192.168.1.0/24 using the **network command.**
- R1 introduces into BGP the prefix 192.168.2.0/24 using **redistribution from OSPF.**

The configuration explained here is just the BGP portion.

```
router bgp 1
        neighbor 10.0.0.2 remote-as 2
        network 192.168.1.0 mask 255.255.255.0
        redistribute ospf 1 route-map OSPF_into_BGP

!
route-map OSPF_into_BGP permit 10
 match ip address 1
!
access-list 1 permit 192.168.2.0 0.0.0.255
!
```

With these commands:

- The neighbour 10.0.0.2 in AS2 is specified to be the peer
- The address 10.0.0.2 correspond to the Juniper router J1

In this part:

- The network command is used to introduce the knowledge of the route prefix 192.168.1.0 into BGP so it can be advertised.
- Also, the prefix learned from OSPF is to be advertised by using redistribution from one protocol (OSPF) to another (BGP)

- But not all the prefixes learned via OSPF are to be advertise.
- Only the desired prefixes will be advertised.
- This access list matches only the desired prefix (192.168.2.0/24) .

Figure 3.9

- JUNOS1 (J1) advertises to R1 the prefixes 172.16.1.0/24 and 172.16.2.0/24 only
- J1 introduces into BGP the prefix 172.16.1.0/24 from **directly connected**
- J1 introduces into BGP the prefix 172.16.2.0/24 using **redistribution** from OSPF.

```
set routing-options router-id 10.2.1.1
set routing-options autonomous-system 2

edit protocols bgp group Peer_with_AS1
        set type external;
        set neighbour 10.0.0.1
        set peers-as 1
        set export To_AS1
```

In this part:
- Basic BGP parameters are defined.

In this part:
- A BGP Group contains the conditions to peer with a neighbour AS.
- J1 router is configured to peer with the neighbour R1 in AS1.
- J1 will only advertise the routing information that passes the conditions in the policy **To_AS1**

```
edit policy-options policy-statement To_AS1
        set term 1 from route-filter 172.16.1.0/24 exact
        set then accept
        set term 2 from protocol ospf route-filter 172.16.2.0/24 exact
        set then accept
        set term 3 then reject
```

In this part:
- The policy **To_AS1** specifies that only the routes 172.16.1.0/24 which is directly connected and 172.16.2.0/24 which is learned via OSPF are to be advertised.

Figure 3.10

Once that all the routers have been configured, a BGP conversation is started between the routers located at the borders of the autonomous systems. This conversation is being sniffed with Wireshark and it will be dissected in the next pages to demonstrate the functionalities of the following BGP attributes: ORIGIN, AS_PATH, NEXT_HOP, and MED. These are the four attributes that appear, minimum, in all BGP UPDATES.

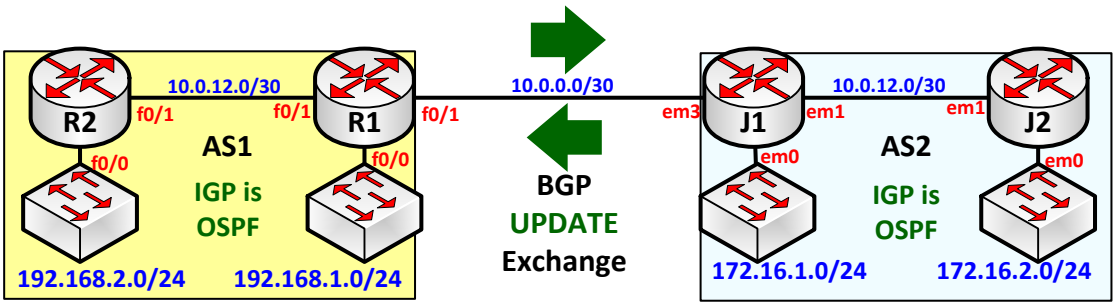


Figure 3.11

The initial observation is done on the UPDATE message sent by Cisco R1 to the Juniper J1.

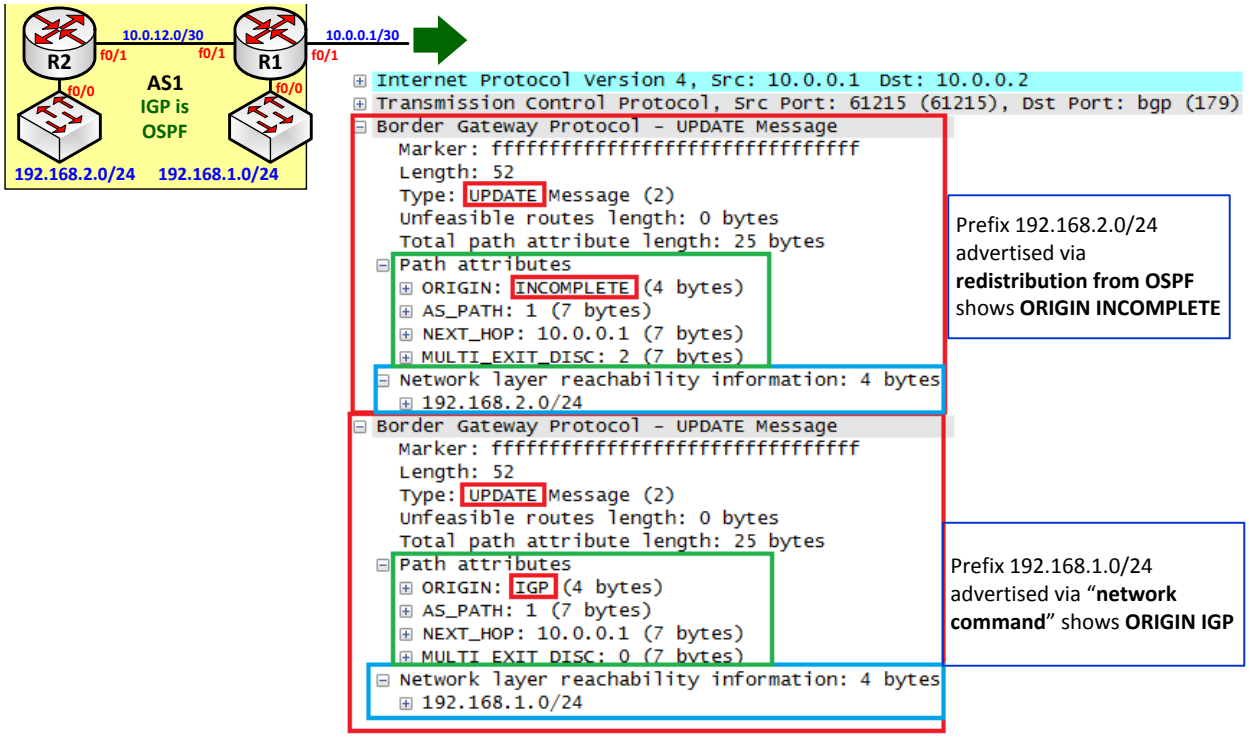


Figure 3.13

The encapsulation shows that an IP packet is sourced from 10.0.0.1 (R1 interface) with a destination 10.0.0.2 (J1 interface) and that this IP packet encapsulates a TCP segment which contains two different UPDATE messages. There are two UPDATES in the TCP payload because there are two different prefixes with different attributes. TCP would try to carry as many UPDATES as they fit in the payload.

Cisco R1 advertises the prefix 192.168.1.0/24 with an IGP Origin and the prefix 192.168.2.0/24 with an INCOMPLETE Origin. Remember from the configuration section earlier that the prefix 192.168.1.0/24 was introduced into BGP via the manual command "network" and that 192.168.2.0/24 was introduced via redistribution from OSPF. This is the detailed information of the ORIGIN Attribute that accompanies prefix 192.168.2.0/24.

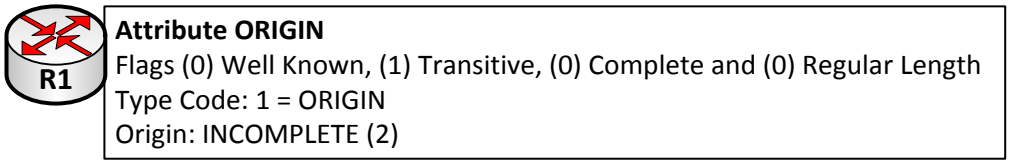


Figure 3.14

The observation continues with the Juniper router implementation of BGP ORIGIN attribute. In this case, Juniper advertises both prefixes 172.16.1.0/24 and 172.16.2.0/24 with IGP Origins.

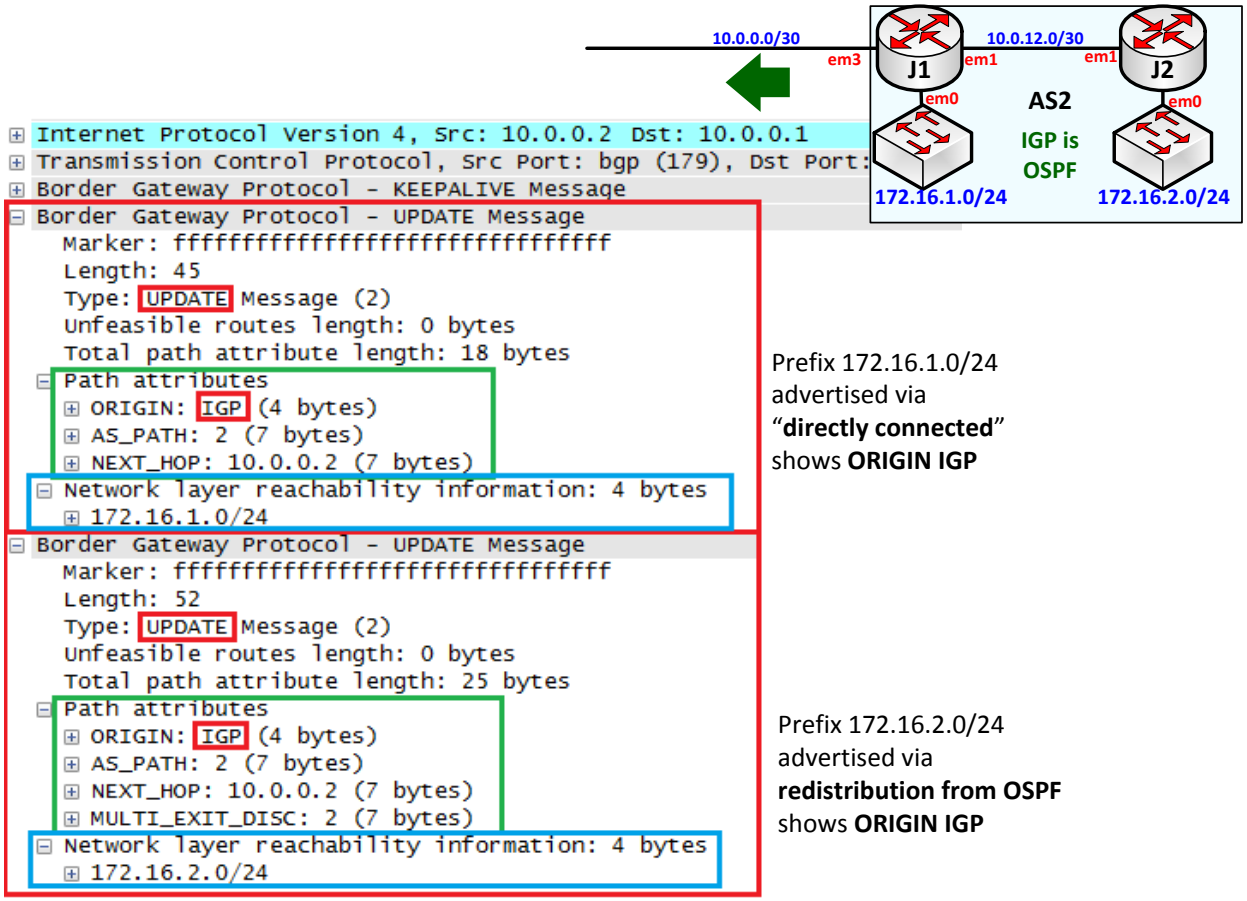


Figure 3.15

JUNOS implementation of BGP advertises both attributes with the Origin IGP. Basically, Juniper routers only advertise routes with ORIGIN IGP while Cisco IOS only advertises with ORIGIN IGP those routes that are injected into BGP with the use of the "network" command. In Cisco IOS, the routes that come via redistribution from an IGP are advertised with an INCOMPLETE Origin. When an incomplete origin appears in a BGP table most likely points out to a route advertised by a Cisco router using redistribution from an IGP. It is hard to know why these vendors had different interpretations for this BGP attribute. An educated guess is that introducing a network via manual command requires a conscious decision by the administrator so the route should have a higher level of confidence than a route that was injected from redistribution.

BGP Attribute AS_PATH

The AS_PATH attribute contains the numbers of the autonomous systems in the path to reach a route destination. The AS_PATH attribute is a critical piece in the functioning of BGP for it provides a way to determine the shortest path to destinations and to break routing loops.

Each BGP border router attaches its own AS number to the prefixes being advertised. In effect, router R1 inserts its AS number (AS #1) into the AS_PATH attribute of the routes that is advertising to J1. The details of an AS_PATH have been dissected from an UPDATE next:

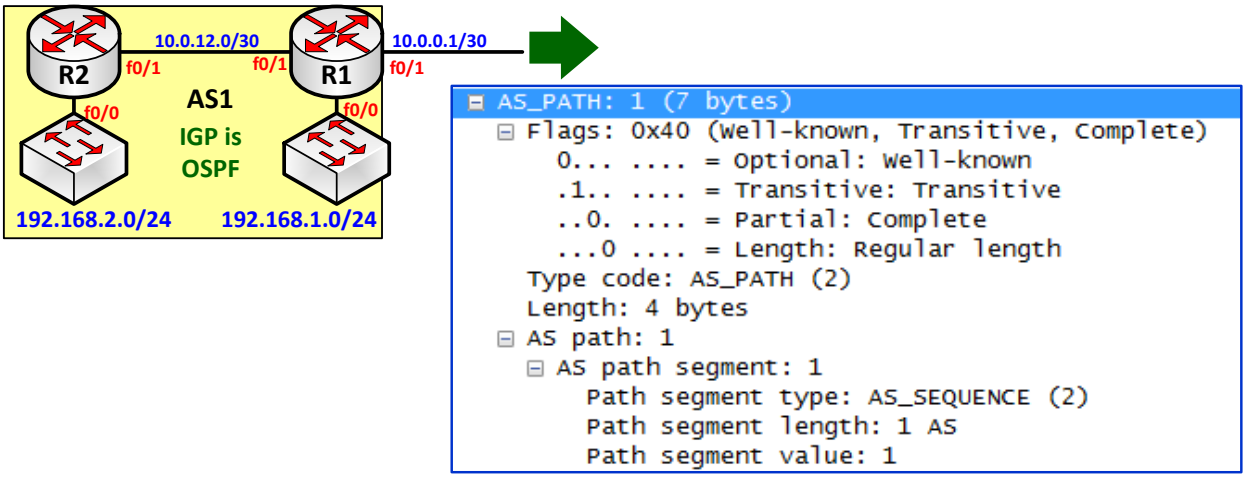


Figure 3.16

The attribute AS_PATH is well-known so all BGP router have to recognize it and it is also mandatory. AS_PATH is transitive as it is carried from AS to AS when BGP UPDATEs propagate the routing information. Each AS adds its own autonomous system number when the border routers advertise the route prefix. This is called an **AS_SEQUENCE**. There is another type of AS_PATH description called **AS_SET** which will be explained later. The next diagram shows the standard fields of the AS_PATH attribute:

Flags	ATTRIBUTE TYPE	Length of AS_PATH	Path segment type	Length of this path segment	Path Segment (2 Bytes)
0 = well known 1 = transitive 0 = complete 0 = regular length 0 = reserved 0 = reserved 0 = reserved 0 = reserved	code # 2 AS_PATH	4 bytes space to describe length	AS_SET (1) AS_SEQUENCE (2)	Number of Autonomous Systems in the path description	The list with the AS numbers in the PATH

Figure 3.17

The length of the attribute simply states how many AS numbers are described. To better illustrate this point, observe the following UPDATE from a different network shown in the next diagram. In this case, several AS numbers are described in the AS_PATH attribute. Router R3 is sending an UPDATE message to router R4 to advertise the prefix 192.168.1.0/24 located in AS1. In turn, router R3 learned this prefix from router R2.

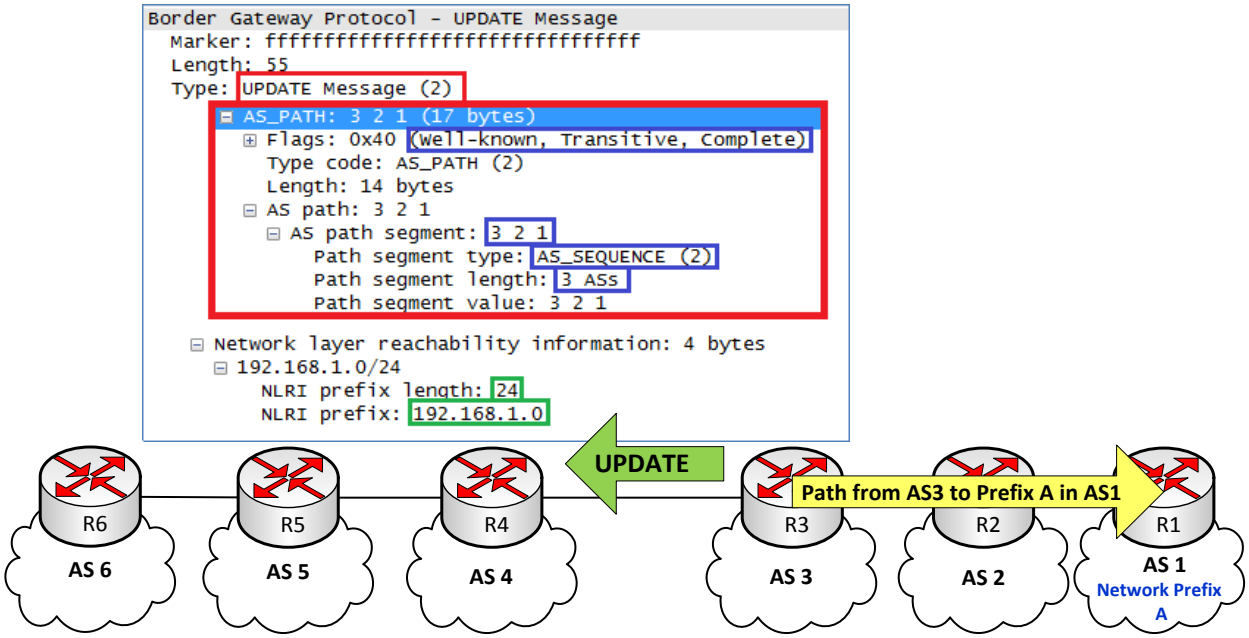


Figure 3.18

Continuing with this diagram, Router R4 sends an UPDATE to router R5 in AS5. Most of the details of the UPDATE message have been omitted except the part that contains the AS_PATH attribute which is rather expanded, showing the path from AS4 to prefix A in AS1. Two of the AS numbers (four and three) are described in binary detail and added to the illustration for the purpose of this explanation.

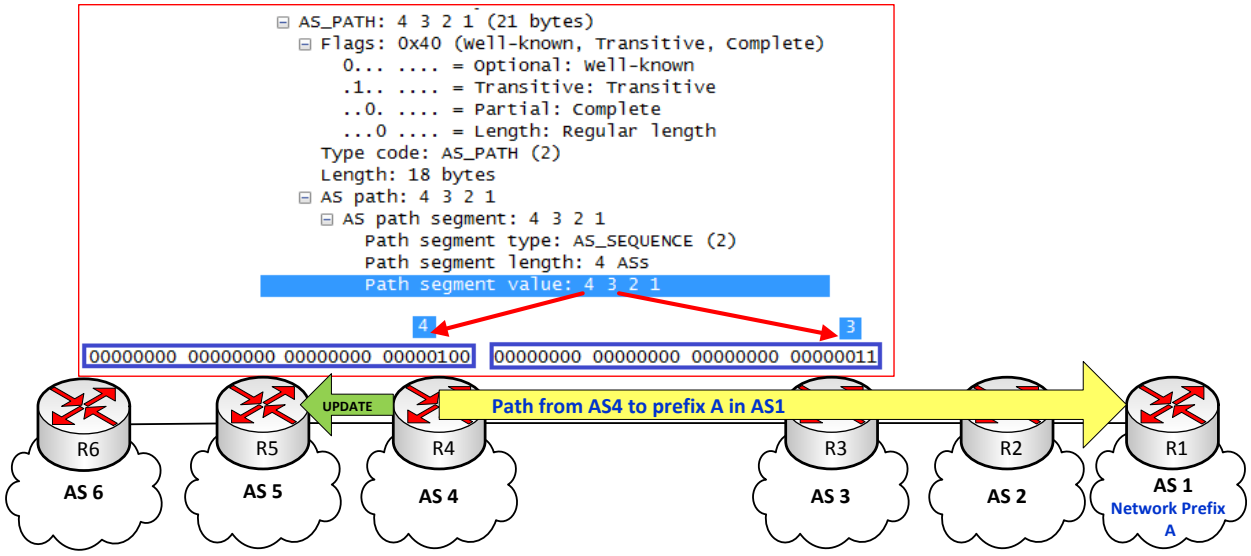


Figure 3.18

The AS_PATH attribute type indicates that this UPDATE contains an AS_SEQUENCE and the Path segment length says that there are four (4) autonomous systems in the sequence. Then the specific AS numbers are listed starting from AS4, then the immediate neighbour to AS4, all the way to the destination.

Also, as observed, each AS is described in a 32 bits field. This is an extended format Autonomous System number that can range from 0 to 2^{32} (more than 4 billions). Previously, the numbering system for BGP ASN ranged from 0 to 2^{16} (65,535) but since the Internet keeps growing and the range was being depleted then the extended format of 32 bits was introduced

Analysis of AS_PATH Attribute

The following network topology is used to further demonstrate the basic mechanics of the AS_PATH attribute. The configuration is just basic BGP by default and the internal details of the autonomous systems are not relevant for this explanation so they are just shown as "clouds". Prefix A (192.168.1.0/24) in Autonomous System 1 is the only network being advertised. The result of the AS_PATH will be observed in router R4 in AS4 (Cisco IOS commands)

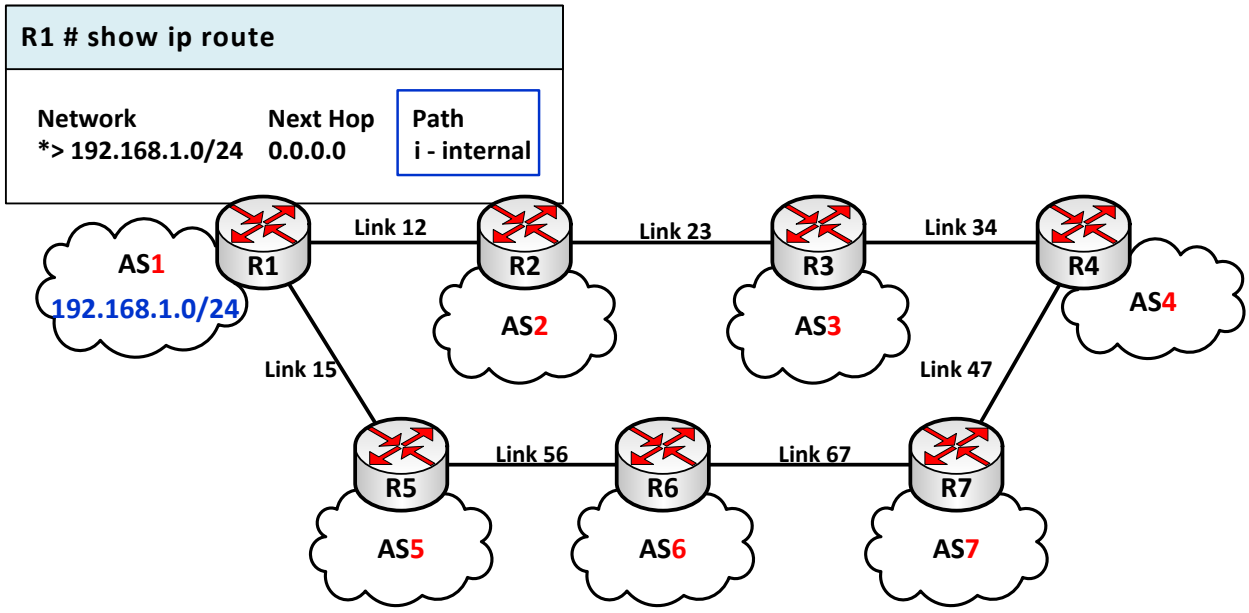


Figure 3.18

The BGP table in R1 shows that the route 192.168.1.0/24 has an **internal PATH** and that makes sense since it is directly connected to router R1. Notice that the Path field does not have AS1 implicitly. Router R1 will write the AS number in the AS_PATH attribute when it creates the UPDATES that would be sent to the neighbours R2 and R5. These are BGP databases for these routers:

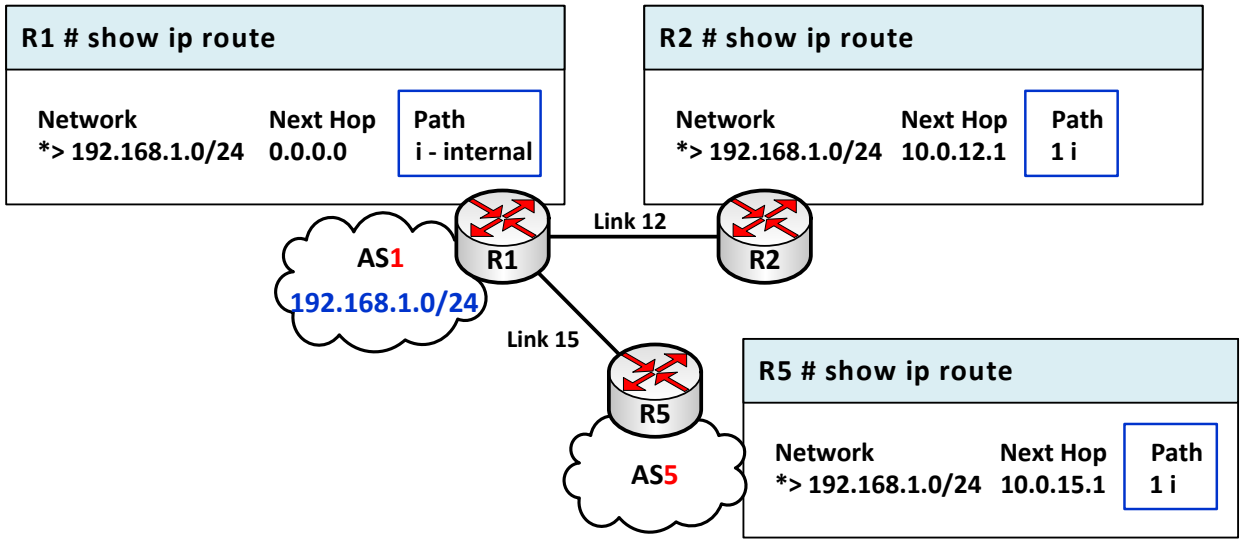


Figure 3.19

Both routers R2 and R5 know that prefix A is located inside autonomous system 1 thanks to the AS_PATH attribute. The only difference between the two routers BGP databases is the next hop address to reach the route. The route propagation process is like this: router R2 receives the UPDATE from router R1 via 10.0.12.1, installs the route in its routing table, and then proceeds to send an UPDATE to its neighbour router R3. Router R5 does the same process, but with its neighbour router R6. When eventually, the propagation of the routing information is complete, the routers databases will show the following information:

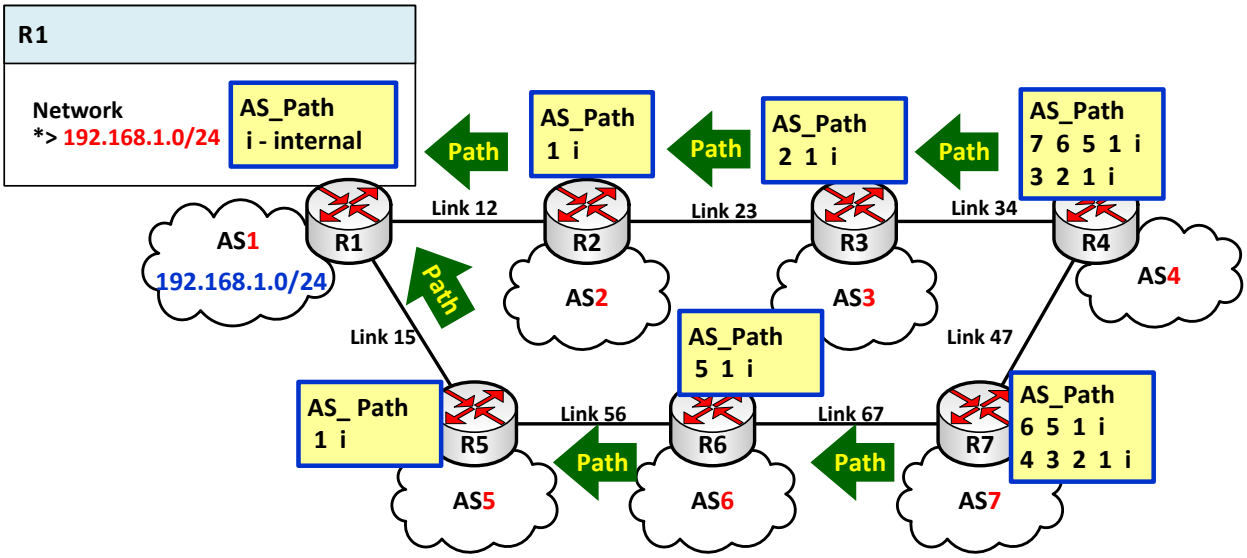


Figure 3.20

Router R3 learns that prefix A is located going the way of AS2, then AS1. Router R6 learns from router R5 that the prefix is via AS5, then AS1. However, an interesting situation happens in R4. In effect, this router has two entries for 192.168.1.0/24 in the BGP database. One of them has a shorter AS_PATH (AS3 AS2 AS1) than the other (AS7 AS6 AS5 AS1) so the best one (indicated with the symbols *>) gets installed into the IP routing table as the following snippet shows:

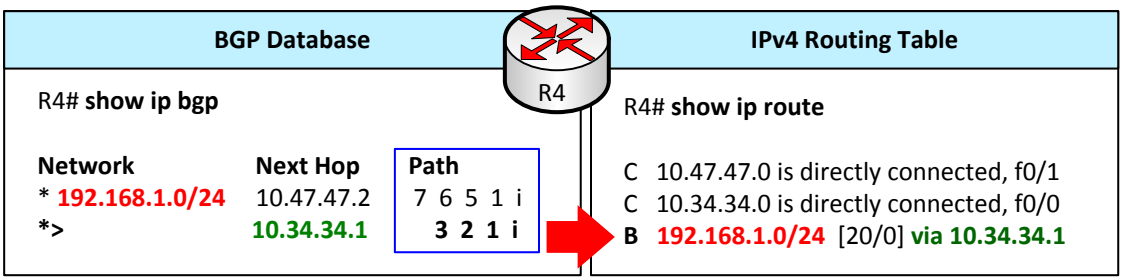


Figure 3.21

Router R7 presents a similar situation with two entries in the BGP database. But notice that only the best one gets promoted into the IP routing table. Now, it is worth to ask for the reason that the other routers do not get two entries in their BGP tables like routers R4 and R7. The answer can be inferred from inspecting the network topology after a bit of physical rearrangement and a review of these basic BGP rules:

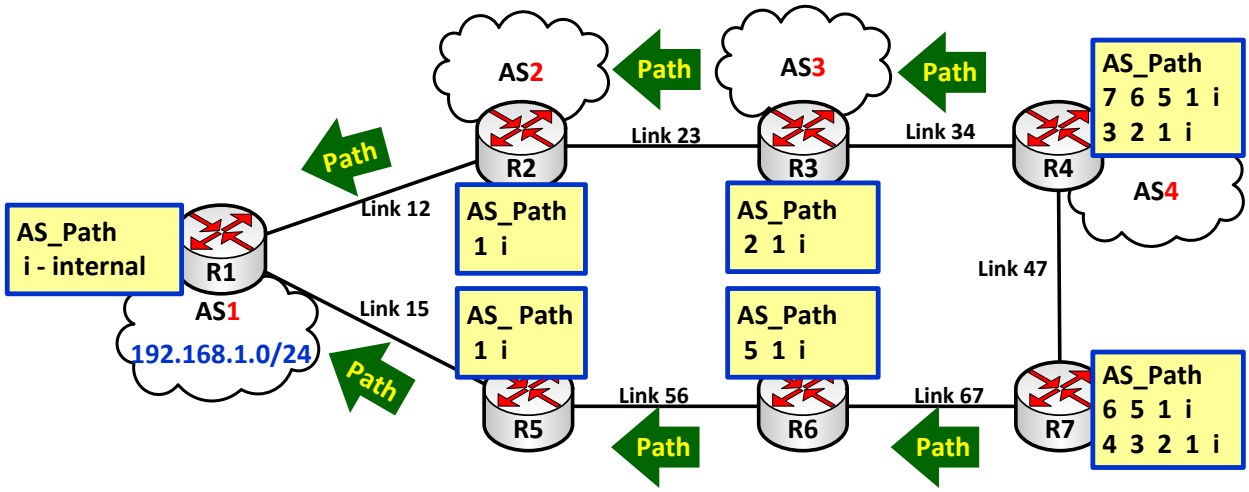


Figure 3.22

- BGP speaking routers keep two set of tables, the BGP database and the IP routing table.
- BGP speaking routers practice split horizon: "do not tell back to a router what that router told you." Furthermore, BGP able routers use split horizon with poison reverse: "that network that you told me about it, cannot be accessed this way".
- BGP speaking routers can only advertise the routes installed in their IP routing tables. "If you cannot reach it yourself, you cannot advertise it."
- BGP is basically a modified distance vector protocol. It is called a **Path Vector Protocol**.

In effect, BGP advertises only* the routes present in the IP routing table. These are precisely the best paths. Since BGP is a distance vector routing protocol it uses "hop count" to calculate the shortest path just like the distance vector routing protocol RIP. The difference is that RIP considers each router as a "hop" while BGP considers each autonomous system as a "hop". It does not matter that there might be 1,000 routers inside the autonomous system, from the view point of BGP each AS is just one "cloud" to be counted in the path to a destination.

*There is a command in JUNOS to advertise inactive routes.

Once stated the previous functioning principles, the analysis of the network can continue. Router R4 concludes that the shortest path to the prefix A is via router R3. After installing such information in its own routing table, R4 sends an UPDATE to R7 saying that prefix A is reachable with an AS_PATH of AS4 AS3 AS2 and then AS1. The following snippet has the details of such UPDATE message. The snippet has been edited to show only the fields related to the AS_PATH attribute.

```
From R4: 10.47.47.1 to R7: 10.47.47.2    BGP UPDATE Message
(Edited for simplicity)
⊟ Path attributes
  ⊟ AS_PATH: 4 3 2 1 (21 bytes)
    ⊞ Flags: 0x40 (Well-known, Transitive, Complete)
      Type code: AS_PATH (2)
      Length: 18 bytes
    ⊞ AS path: 4 3 2 1

  ⊟ Network layer reachability information: 4 bytes
    ⊞ 192.168.1.0/24
```

Figure 3.23

R7 receives the previous UPDATE, however, router R7 has also reached a conclusion that the best path to prefix A is via R6. This path is shorter than the one advertised by R4, so R7 keeps its best path and it also proceeds to advertise to its peer R4 the route A with attribute AS_PATH AS7 AS6 AS5 AS1. They both accept the information from the UPDATES because it complies with all the rules stated above, but the information remains in the BGP database marked as valid (with the symbol *) but only the best paths (marked *>) will go to the IP routing table.

Previously, there was the question of why the other routers do not get two paths to the same destination. The answer is in the way that the mechanics of the BGP engine settles in a converged state. The first BGP rule to be applied here is that Router R4 cannot tell router R3 about the advertisement coming from R7 because that is not the path being used by R4 to reach prefix A. It is simply not the best path installed in the IP routing table. The second BGP rule in use is "split horizon" that R4 cannot tell R3 what R3 already told R4. The same principles apply to the conversation between R7 and R6.

Use of AS_PATH to break routing loops

Finally, the attribute AS_PATH has a critical role for the functioning of BGP. In the diagram below, R1 has been reset and a sequence of events has resulted in an interesting case. Actually, this is a situation difficult to replicate, it just happened by chance. After restarting R1, this router has sent an UPDATE to R5 but for some reason it has delayed the UPDATE due to R2. In the meantime, R5 has propagated the information toward R6 and from there, the chain of events continue all the way to R2. before R1 can directly update R2. So now, R2 sends an UPDATE to R1 saying that 192.168.1.0/24 can be reached via the path AS2, AS3, AS4, AS4, AS7, AS5, and finally AS1 which is technically true but clearly absurd.

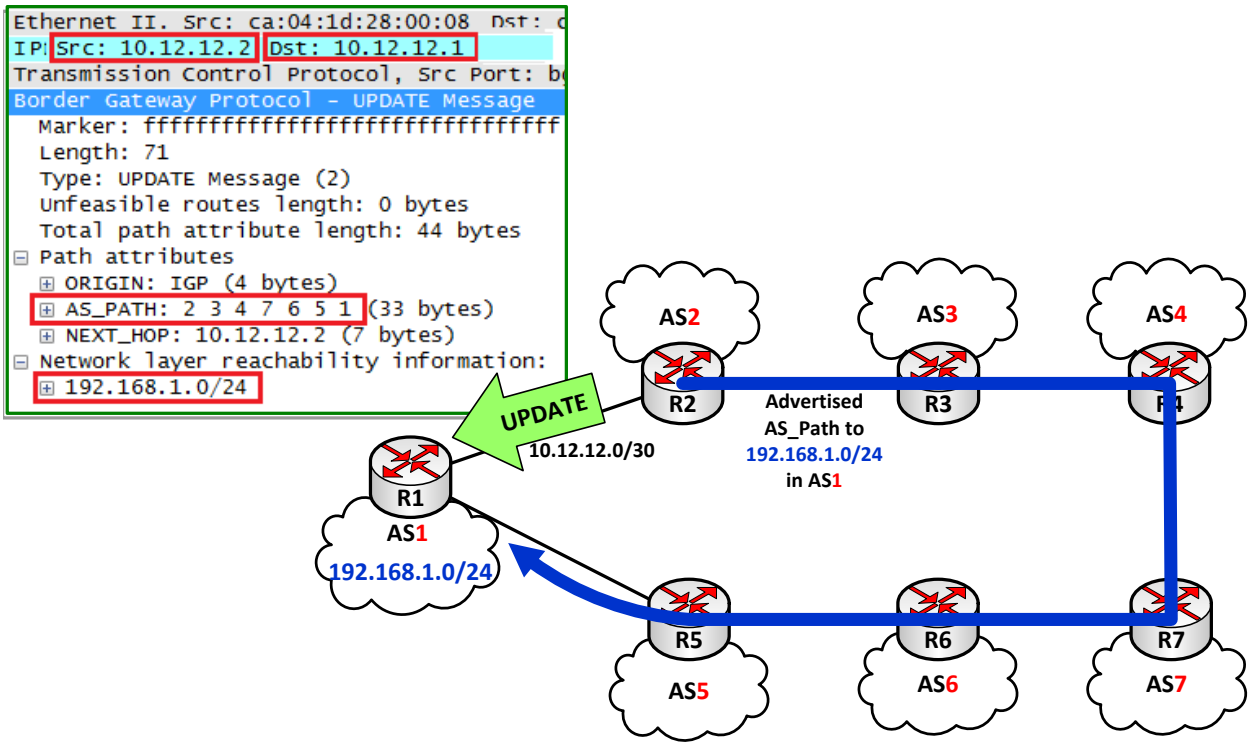


Figure 3.24

In spite of this, when R1 receives the UPDATE, it inspects the AS_PATH attribute, and it realizes that its own AS number is in the advertised path so it proceeds to block this information from being installed in its local BGP table. That is a BGP rule: "**when a BGP router sees its own AS in the AS_PATH, it will ignore the UPDATE**". The process settles down when R1 UPDATES R2 with the correct information and soon the routing knowledge will achieve a converged correct state. This demonstrate how the AS_PATH attribute helps BGP to avoid routing loops.

BGP Attribute NEXT_HOP

The last of the three well-known mandatory attributes is the NEXT_HOP. When a BGP border router prepares an UPDATE, it writes the IP address of the outgoing interface in the value field of the NEXT_HOP attribute. The router that receives the UPDATE is typically directly connected on the same network. For example, router R3 sends an UPDATE to router R4 with the NEXT_HOP set to 10.34.34.1. Since router R4 is connected at 10.34.34.2, it will have no trouble determining that R3 is the NEXT HOP for the routes that advertises. In the snippet, the BGP table has two paths to the destination 192.168.1.0/24, but the best path is only found via 10.34.34.1.

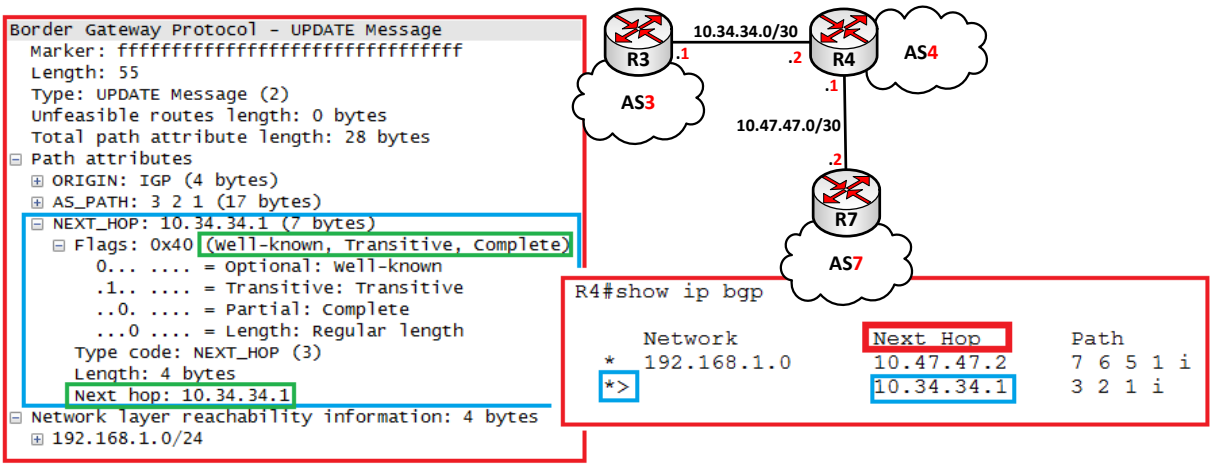


Figure 3.25

The use of the BGP attribute NEXT_HOP follows a few rules:

- The NEXT_HOP must be reachable in order for a router to accept and use the routes in the UPDATE message.
- If the NEXT_HOP is not present in the IP routing table, the routes in the UPDATE related to such NEXT_HOP are dropped.

The lack of understanding of the NEXT_HOP attribute is a major cause of many configuration problems when BGP is ran inside autonomous systems. When BGP is used to carry routing information across an autonomous system it is called internal BGP or iBGP. This is the case:

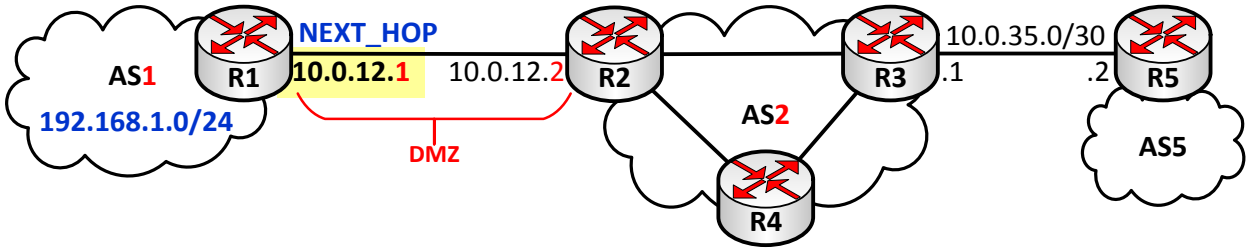


Figure 3.26

Autonomous system AS2 provides transit to the IP traffic between AS1 and AS5. To do so, the BGP UPDATES must carry the routing information across AS2. This is done using BGP but with a few configuration changes including a different treatment of the NEXT_HOP attribute.

When router R1 sends an UPDATE to router R2, it writes 10.0.12.1 in the NEXT_HOP attribute which correspond to the interface sourcing the message. Router R2 installs the routes associated to such NEXT_HOP in its BGP database. This is not a problem for R2 because it is directly connected to the same physical link. Next, R2 should advertise the routes to R3 and R4, but the BGP default behaviour is that:

- a BGP speaking border router introduces the external routes with the NEXT_HOP unchanged to internal BGP routers.

Router R3 receives the UPDATE with the routes associated to NEXT_HOP 10.0.12.1 and the problem begins. R3 does not know where 10.0.12.0/30 is located. This router only knows its directly connected networks plus the routes advertised by an IGP inside the AS2. The link 10.0.12.0/30 is called a DMZ (demilitarized zone) and it does not belong necessarily to AS2. Since another BGP rule says that **routes with unreachable NEXT_HOP should not be accepted**, then R3 does not install the information in the IP routing table. The same situation occurs with the routing updates coming from R5 into R3. In this case, R2 has no clue where 10.0.45.2 is located.

There are several solutions to this problem. One type of solution is that some routing information can be introduced into AS2 about the DMZ links. The most common solution is that the NEXT_HOP gets replaced with reachable information. This topic will be better understood when internal BGP is studied in a later chapter.

Chapter Summary

Attributes are the descriptors of a BGP object. Consequently, knowing the BGP attributes and how to use them is fundamental for the deployment and troubleshooting of autonomous systems. In fact, BGP policies typically match three factors which are the IP prefix, the mask, and the attributes.

In this chapter, the structure of the BGP attribute has been dissected and explained at the field and bit level. Attributes comply with the TLV design paradigm where a Type identifies the object, a Length field points to the total size in bytes of the object and finally, the Value field contains the actual object.

There are many BGP attributes, both mandatory and optional. In this chapter, the three main attributes that appear in all BGP implementation have been described. These mandatory attributes are the Origin of the route, the AS_PATH to get to the route and the NEXT_HOP that points to the route. The rest of the attributes will be studied in the next chapter as they are applied to particular situations.

Exercise

Configuration of network for practice and analysis

- Implement the network topology. All the configurations are included below.
- Sniff the traffic on 10.12.12.0/30 between R1 and R2.
- Observe and analyze the BGP OPEN message.
- Analyze the BGP attributes present in the BGP UPDATE message.
- Analyze the IP routing tables and BGP database of each router.

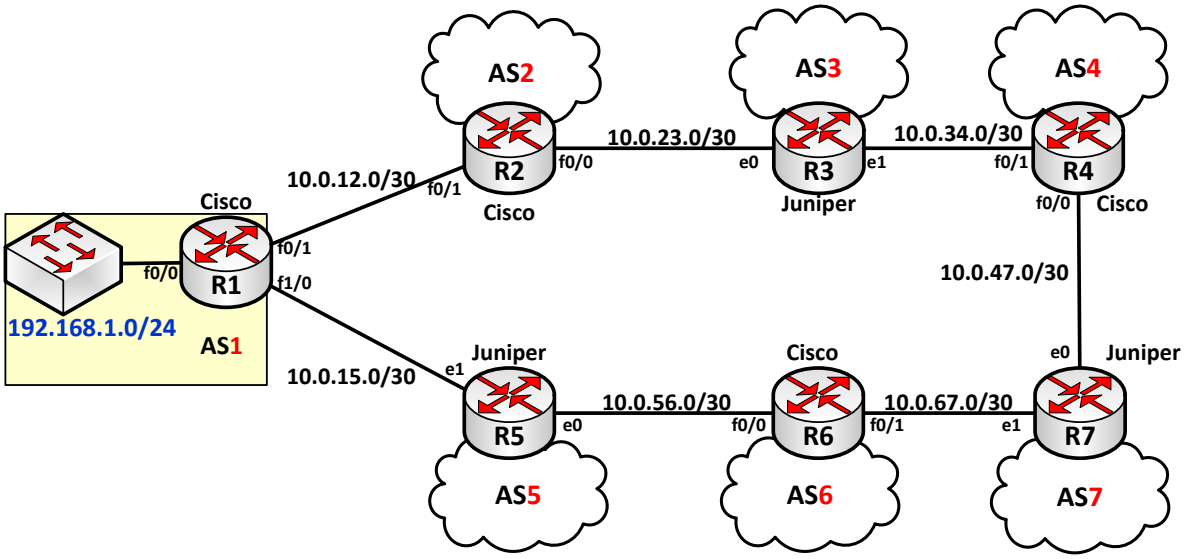
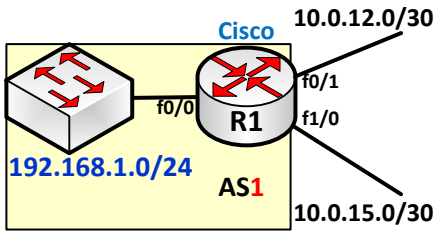


Figure 3.27



router bgp 1
neighbor 10.0.12.2 remote-as 2
neighbor 10.0.15.2 remote-as 7
network 192.168.1.0

hostname R1

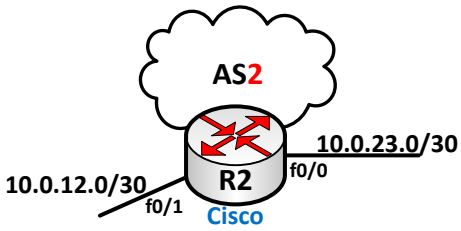
interface lo0
ip address 10.1.1.1 255.255.255.255

interface f0/0
ip address 192.168.1.1 255.255.255.0
no shutdown

interface f0/1
ip address 10.0.12.1 255.255.255.252
no shutdown

interface f1/0
ip address 10.0.15.1 255.255.255.252
no shutdown

Figure 3.28



router bgp 2
neighbor 10.0.12.2 remote-as 1
neighbor 10.0.23.2 remote-as 3

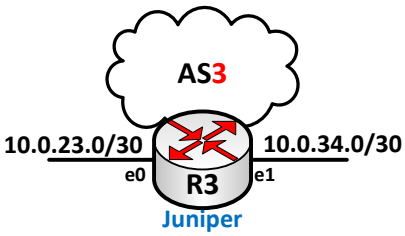
hostname R2

interface lo0
ip address 10.2.2.2 255.255.255.255

interface f0/0
ip address 10.0.23.1 255.255.255.0
no shutdown

interface f0/1
ip address 10.0.12.2 255.255.255.252
no shutdown

Figure 3.29



R3 – Juniper JUNOS

set system **host-name** R3

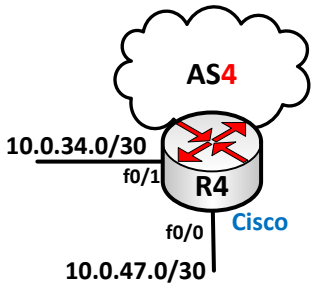
set **interfaces lo0** unit 0 family inet address 10.3.3.3/32
set **interfaces em0** unit 0 family inet address 10.0.23.2/30
set **interfaces em1** unit 0 family inet address 10.0.34.1/30

set routing-options **router-id** 10.3.3.3
set routing-options **autonomous-system** 3

set **protocols bgp** group PEERS type external
set **protocols bgp** group PEERS neighbor 10.0.23.1 peer-as 2
set **protocols bgp** group PEERS neighbor 10.0.34.2 peer-as 4

Figure 3.21

```
hostname R4

interface lo0
ip address 10.4.4.4 255.255.255.255

interface f0/0
ip address 10.0.47.1  255.255.255.0
no shutdown

interface f0/1
ip address 10.0.34.2 255.255.255.252
no shutdown
```




```
router bgp 4
neighbor 10.0.34.1 remote-as 3
neighbor 10.0.47.2 remote-as 7
```

Figure 3.30

R7 – Juniper JUNOS

```
set system host-name R7
set interfaces lo0 unit 0 family inet address 10.7.7.7/32
set interfaces em0 unit 0 family inet address 10.0.47.2/30
set interfaces em1 unit 0 family inet address 10.0.67.2/30

set routing-options router-id 10.7.7.7
set routing-options autonomous-system 7

set protocols bgp group PEERS type external
set protocols bgp group PEERS neighbor 10.0.47.1 peer-as 4
set protocols bgp group PEERS neighbor 10.0.67.1 peer-as 6
```

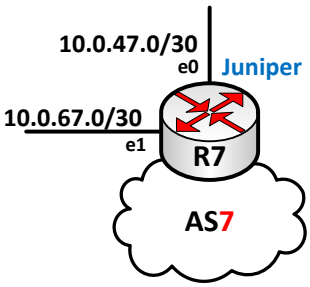
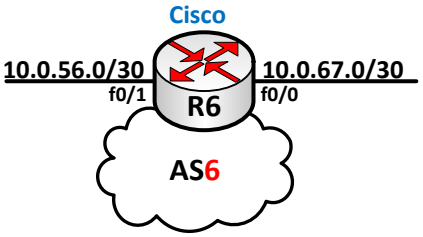


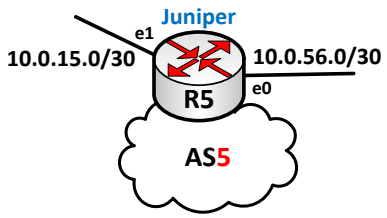
Figure 3.31




```
router bgp 6
neighbor 10.0.56.1 remote-as 5
neighbor 10.0.67.1 remote-as7
```

```
hostname R6

interface lo0
ip address 10.6.6.6 255.255.255.255

interface f0/0
ip address 10.0.67.1  255.255.255.0
no shutdown

interface f0/1
ip address 10.0.56.2 255.255.255.252
no shutdown
```

Figure 3.32



R5 – Juniper JUNOS
set system **host-name** R5
set **interfaces em0** unit 0 family inet address 10.0.56.1/30
set **interfaces em1** unit 0 family inet address 10.0.15.2/30
set **interfaces lo0** unit 0 family inet address 10.5.5.5/32

set routing-options **router-id** 10.5.5.5
set routing-options **autonomous-system** 5

set **protocols bgp** group PEERS type external
set **protocols bgp** group PEERS neighbor 10.0.15.1 peer-as 1
set **protocols bgp** group PEERS neighbor 10.0.56.2 peer-as 6

Figure 3.33

Chapter References

[1] K. Lougheed. Y. Rekhter. (1991). RFC-1267. A Border Gateway Protocol 3 (BGP-3).

http://www.rfc-base.org/txt/rfc-1267.txt

[2] Y. Rekhter, T. Li. (1995). RFC 1771. A Border Gateway Protocol (BGP-4).
https://tools.ietf.org/html/rfc1771

[3] Y. Rekhter, T. Li, S. Hares. (2006). RFC 4271. A Border Gateway Protocol (BGP-4).
https://tools.ietf.org/html/rfc4271

[4] IANA. Border Gateway Parameters. (2016). http://www.iana.org/assignments/bgp-parameters/bgp-parameters.xhtml

[5] B. Manning. (1997) RFC 2042. Registering New BGP Attribute Types.
https://tools.ietf.org/html/rfc2042

CHAPTER 4

BGP inside the Autonomous System

BGP is the routing protocol that holds together the autonomous systems in the Internet. Many of these AS belong to very large Internet providers that can span throughout continents and beyond. Other smaller organizations just use the access services offered by the carriers to get passage to other destinations. But before the IP traffic can even begin to flow, the knowledge about the locations of the IP routes has to propagate throughout the transit autonomous systems and that is done with BGP. Consequently, there are two basic deployment profiles for BGP: externally among AS and internally the AS. This chapter deals with internal BGP or iBGP.

Fundamentals of iBGP

The main idea behind the operation of iBGP is to peer inside the autonomous system. Before getting into the details concerning the operation of iBGP, it is necessary to understand the need for using BGP internally. In the following diagram:

- Routers R2a peers with R1 using BGP.
- Routers R2b peers with R3 using BGP.
- R2a and R2b run an IGP internally AS2.

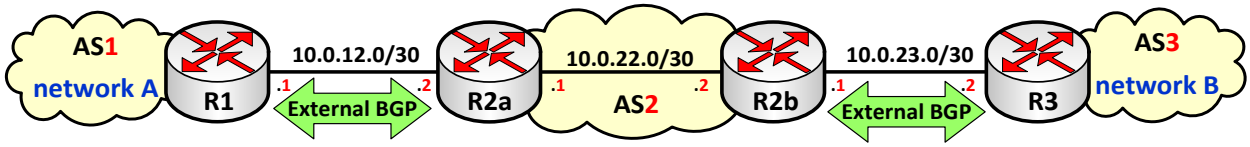


Figure 4.1

As a result, R2a learns from R1 about network N1 in AS1. On the other side of AS2, router R2b learns from R3 about network N3 inside AS3. All of this learning is done via external BGP or default BGP. Then, these AS2's border routers install the IP prefixes in their routing tables which are summarized in the following animated diagram.

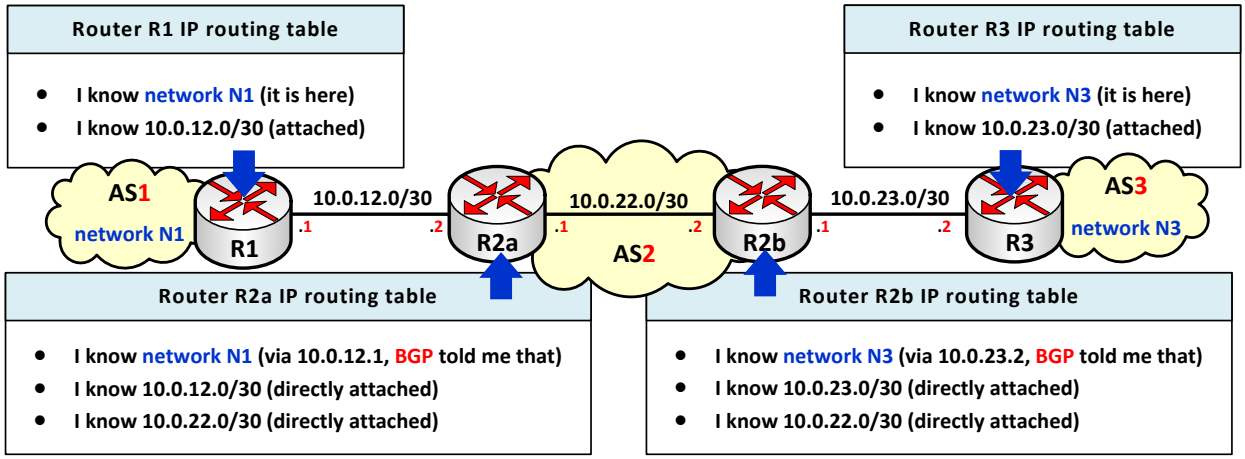


Figure 4.2

The problem is that neither R1 nor R3 learn the network destinations beyond AS2. The disconnection is located in AS2 since routers R2a and R2b do not propagate the information received by the BGP neighbours. Basically, R2a and R2b need to have a conversation about N1 and N3 and that is not happening. Since they are already talking to each other using an IGP, it should be possible to translate (**redistribute**) the external information from BGP into the IGP. As a proof of concept, redistribution of routing information between BGP into an IGP will be done and the effect will be analyzed next. For this explanation, OSPF has been chosen to be the IGP, but it could be any other protocol.

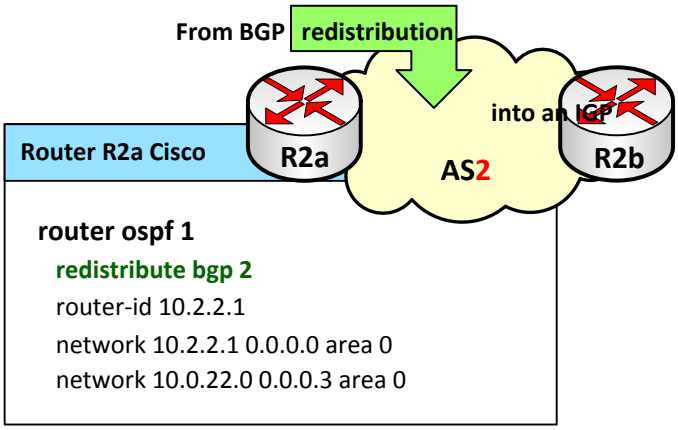


Figure 4.3

The IP routing table of R2b will show if the routing information has been received.

R2b IP routing table

```
R2b# show ip route
C     10.2.2.2/32 is directly connected, Loopback0
C     10.0.23.0/30 is directly connected, f0/1
C     10.0.22.0.0/30 is directly connected, f0/0
O     10.2.2.1/32 [110/2] via 10.0.22.1, f0/0
O  E2  192.168.1.0/24  [110/1] via 10.0.22.1, f0/0
B    192.168.3.0/24 [20/0] via 10.0.23.2
```

Table 4.1

The prefix 192.168.1.0/24 is present in R2b's routing table, so far, so good. Next, in the same router, IGP is redistributed into BGP. This will allow the routing information about network N1 to continue toward R3-AS3. The keyword OSPF type external is used to match only the OSPF type E2 in the previous routing table, thus avoiding redistribution of other prefixes inside AS2.

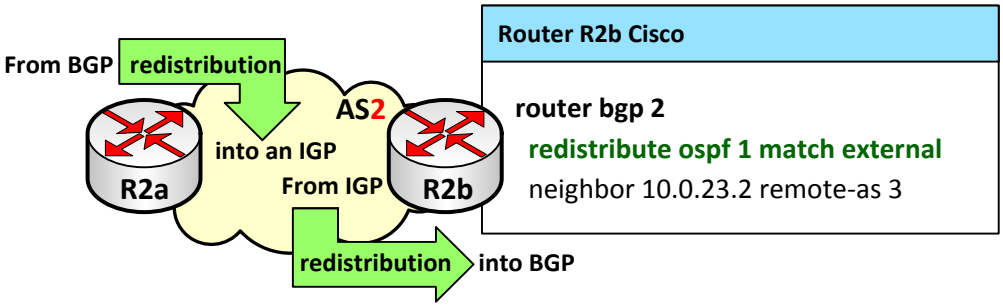


Figure 4.3

The effect is observed in R2b's BGP database:

```
R2b#show ip bgp
  Network          Next Hop      Metric  LocPrf  Weight   Path
*> 192.168.1.0     10.0.22.1     1               32768    ?
*> 192.168.3.0     10.0.23.2     0               0        3 i
```

Table 4.2

The prefix 192.168.1.0/24 does appear in the BGP table, but it is not right. The AS_PATH attribute has been lost during the process and that is the main problem with redistribution. The IGP protocols can only carry the prefix, mask and metric, but they do not have the capability to carry the BGP attributes. Even worse, router R3 have learned that 192.168.1.0/24 is located inside AS2 which is clearly false. A look inside R3 proves this point:

R3# show ip bgp					
Network	**Next Hop**	**Metric**	**LocPrf**	**Weight**	**Path**
***> 192.168.1.0**	**10.0.23.1**	**1**		**0**	**2 ?**
*> 192.168.3.0	0.0.0.0	0		32768	i

Table 4.3

This confirms that redistribution is not the solution to get the routing information through the AS. Furthermore, there are several other problems that arise by using redistribution. For networks in AS1 and AS3 to be reachable both ways, 192.168.3.0 would also need to be redistributed from BGP into OSPF in R2b and then from OSPF into BGP in R2a. Once this is finished, a chicken and egg problem appears because of the mutual redistribution between BGP and the IGP. So much that BGP detects the anomaly and it proceeds to label the entries with a "RIB failure" label in its database. Fixing this situation requires so much additional configuration that is not even worth it.

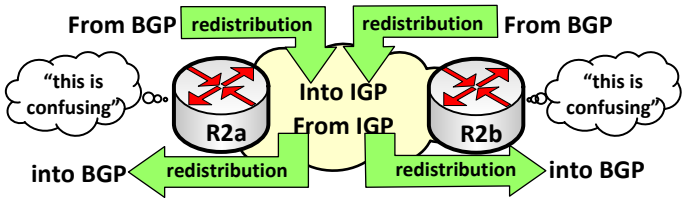


Figure 4.4

The two routers in AS2 are not aware that they belong to the same AS since they do not exchange routing information via BGP, although they do with their external peers, this effectively creates two AS2 which is illogical and wrong too.

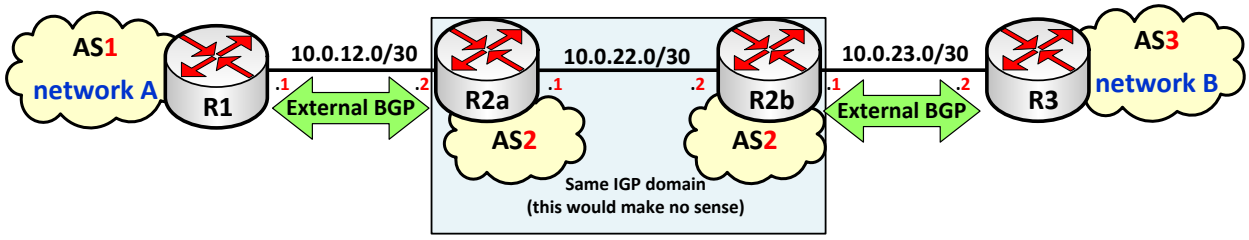


Figure 4.5

There are many other problems associated with BGP-IGP redistribution, but this example has exposed the main reason why using an IGP to carry external routing information does not work. Instead, BGP must be used to carry the external routing information across an autonomous system. It is time to deploy BGP internally.

Internal BGP (iBGP)

An autonomous system that provides transit to IP traffic must implement an IGP to build up the internal IP connectivity and additionally, the AS border routers (ASBR) must speak BGP internally to carry across the AS the routing updates about external routes. It is only when iBGP is implemented that the BGP attributes are preserved.

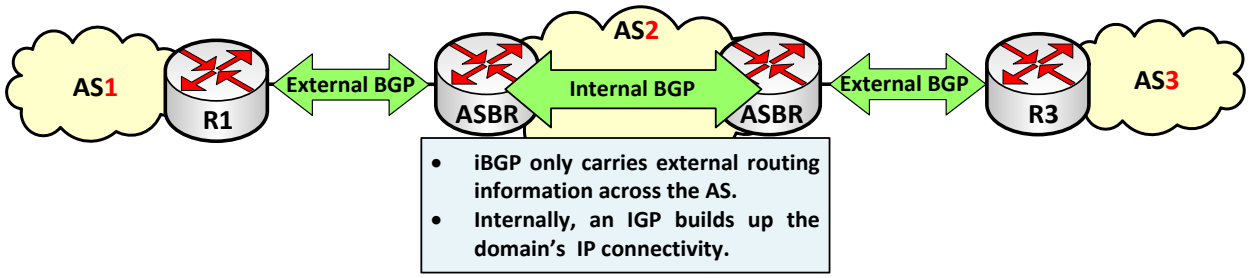


Figure 4.4

There is another reason to speak iBGP inside an AS which is to provide some routers with a view of the external routes. For example the internal router in the diagram below is not an ASBR, but since it provides a potential redundancy path for traffic, it needs to have the knowledge of where the external routes are located.

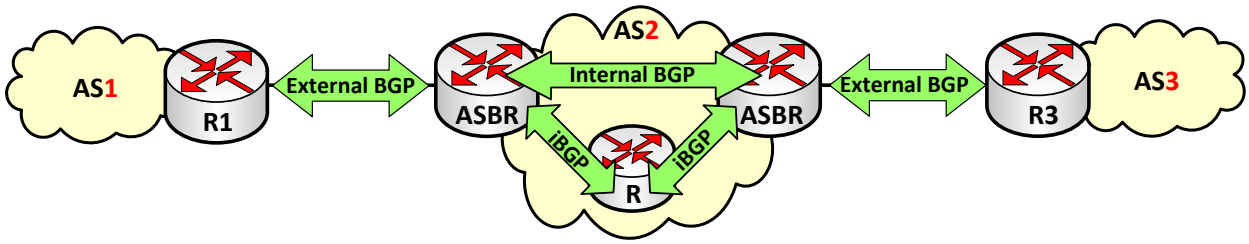


Figure 4.5

It is necessary to emphasize that iBGP does not replace the IGP nor is it meant to work as an IGP. An Internal Gateway Protocol (OSPF, IS-IS, EIGRP, or RIP) is still required inside the autonomous system. iBGP is only used to convey the BGP external routes knowledge across the AS. Internal BGP is the same BGP protocol but with some of its rules relaxed or modified. The functional features of BGP that require special attention when deploying iBGP are:

- the reachability of external routes,
- the NEXT_HOP attribute,
- and the synchronization rule.

They will be explained using the following simple scenario:

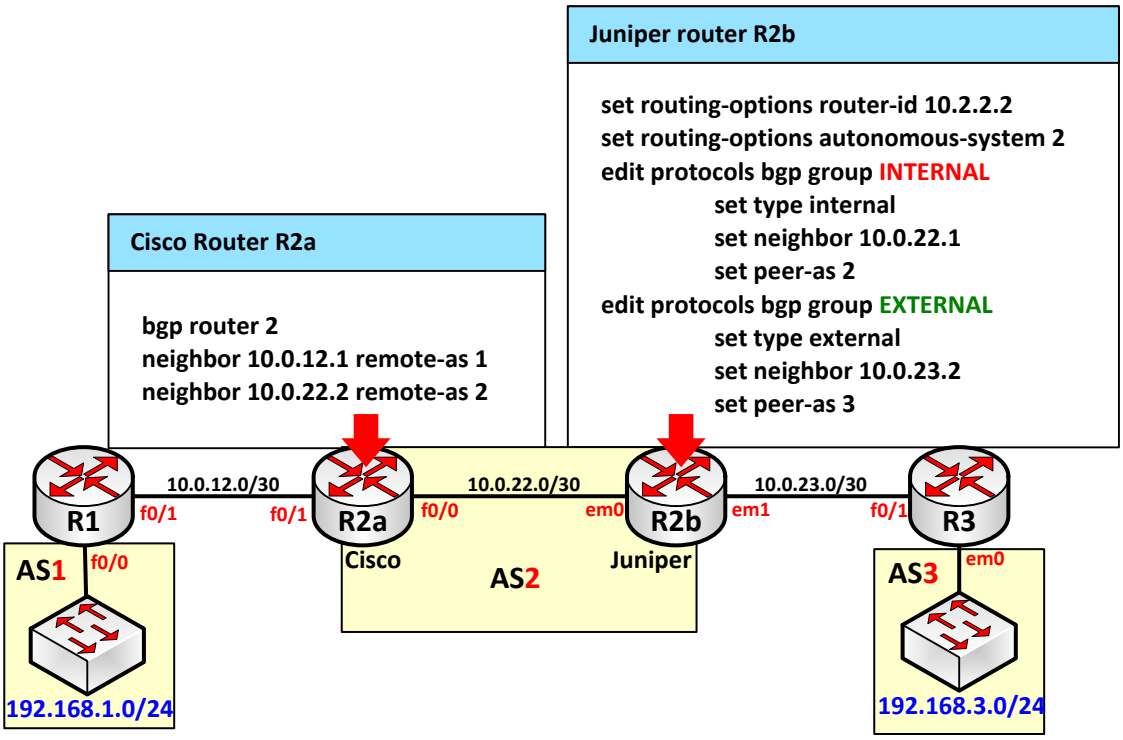
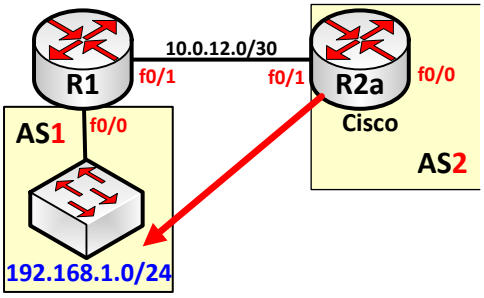


Figure 4.6

Routers R2a (Cisco) and R2b (Juniper) are configured to belong to the same AS2 so they will be speaking iBGP between themselves. The administrator configures the same peer AS number and when the two routers acquire a peer adjacency, they exchange OPEN messages confirming that they are neighbours in the same autonomous system. So they proceed to exchange UPDATES with the routing information. Router R2a knows the following in its routing table:




```
R2a#show ip route

O    10.2.2.2/32 [110/1] via 10.20.20.2, f0/0
C    10.0.22.0/30 is directly connected f0/0
C    10.0.12.0/30 is directly connected, f0/1
C    10.2.2.1/32 is directly connected, lo0
B    192.168.1.0/24 [20/0] via 10.0.12.1
```

Table 4.4 and Figure 4.7

Router R2b knows the following in its routing table:

R2b> show route terse

A	Destination	Protocol	Preference	Next hop	AS path
*	10.2.2.1/32	OSPF	10	>10.0.22.1	
*	10.2.2.2/32	DIRECT	0	>lo0.0	
*	10.0.22.0/30	DIRECT	0	>em0.0	
*	10.0.23.0/30	DIRECT	0	>em1.0	
*	192.168.3.0/24	BGP	170	>10.0.23.2	3 i

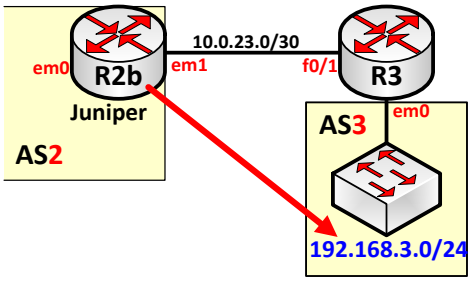


Table 4.5 and Figure 4.8

Enabling iBGP has not accomplished anything. R2a does not have 192.168.3.0/24 in its routing table and R2b does not have 192.168.1.0/24. Something seems to be blocking the propagation of routing knowledge; a look into the BGP table helps to understand the process.

R2a#show ip bgp (edited for simplicity)

Network	Next Hop	AS Path
*> 192.168.1.0	10.0.12.1	1 i
* i192.168.3.0	10.0.23.2	3 i

Table 4.6

Router R2a does have both entries in its BGP database, which implies that it has received the UPDATE about 192.168.3.0/24 from R2b. However, for some reason, the prefix is not promoted into the IP routing table and it remains valid (* symbol) but not used. A look into R2b is needed.

felix@R2b> show route protocol bgp all terse

(Edited for simplicity)

A	Destination	P Prf	Next hop	AS path
	192.168.1.0/24	B 170	Unusable	1 i
*	192.168.3.0/24	B 170	>10.0.23.2	i

Table 4.7

The table clearly says that the **Next Hop** to 192.168.1.0/24 is **unusable.** That would be a good reason why the prefix does not get installed into the IP routing table, but why is the next hop unusable? The following diagram pieces together the elements to solve this puzzle.

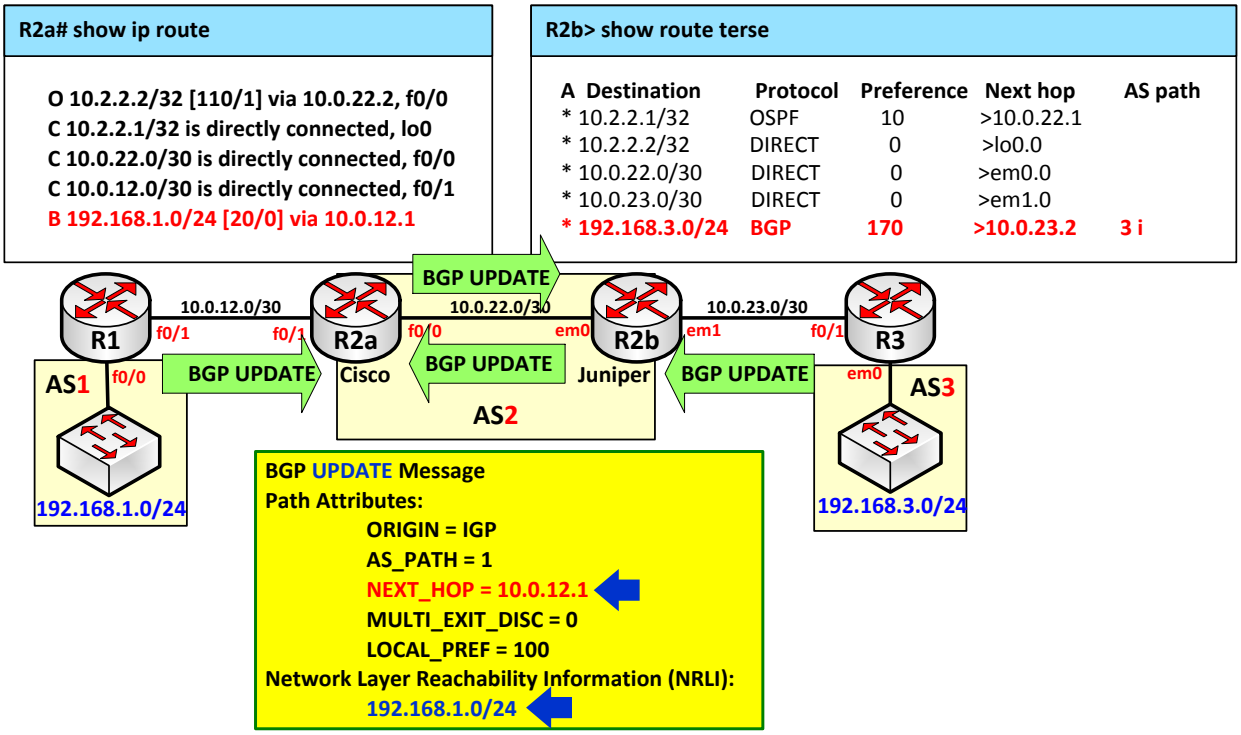


Figure 4.9

The diagram shows R2a's routing table and a sniffed UPDATE sent from R2a to R2b. The body of the UPDATE shows that 192.168.1.0/24 has a NEXT_HOP attribute of 10.0.12.1. The routing table shows that R2a knows 10.0.12.1; of course, it is a directly connected link. But R2b's routing table shows no information about 10.0.12.0/30; simply put, R2b has no clue where to find the NEXT_HOP destination. Consequently, R2b receives the UPDATE from R2a, installs the entry for 192.168.1.0/24 in the BGP table, but it will not use the destination because it does not know where the NEXT_HOP is located. The same situation occurs in R2a regarding 192.168.3.0/24 with respect the next hop 10.0.23.2. in summary, BGP cannot offer the prefix to IP because of the BGP rule that says: "if you cannot reach it, you cannot use it, and you cannot advertise it." BGP will not advertise a black-hole. This is the BGP mechanics acting in a protection role.

AS border routers introduce into iBGP by default, the external routes with the NEXT_HOP attribute unchanged. This is an issue that can be solved in different ways:

- The administrator manually adds a static entry into the routing table about the external NEXT_HOP.
- The administrator enables the IGP in the external link but as a passive interface. Under this condition, the ASBR advertises the external next hop into the IGP without being able to join another IGP domain by accident. This has to be done carefully.
- The most common practice is to turn on a knob in the BGP configuration that allows the ASBR to overwrite the NEXT_HOP address with its own. This is the best practice solution that will be implemented next.

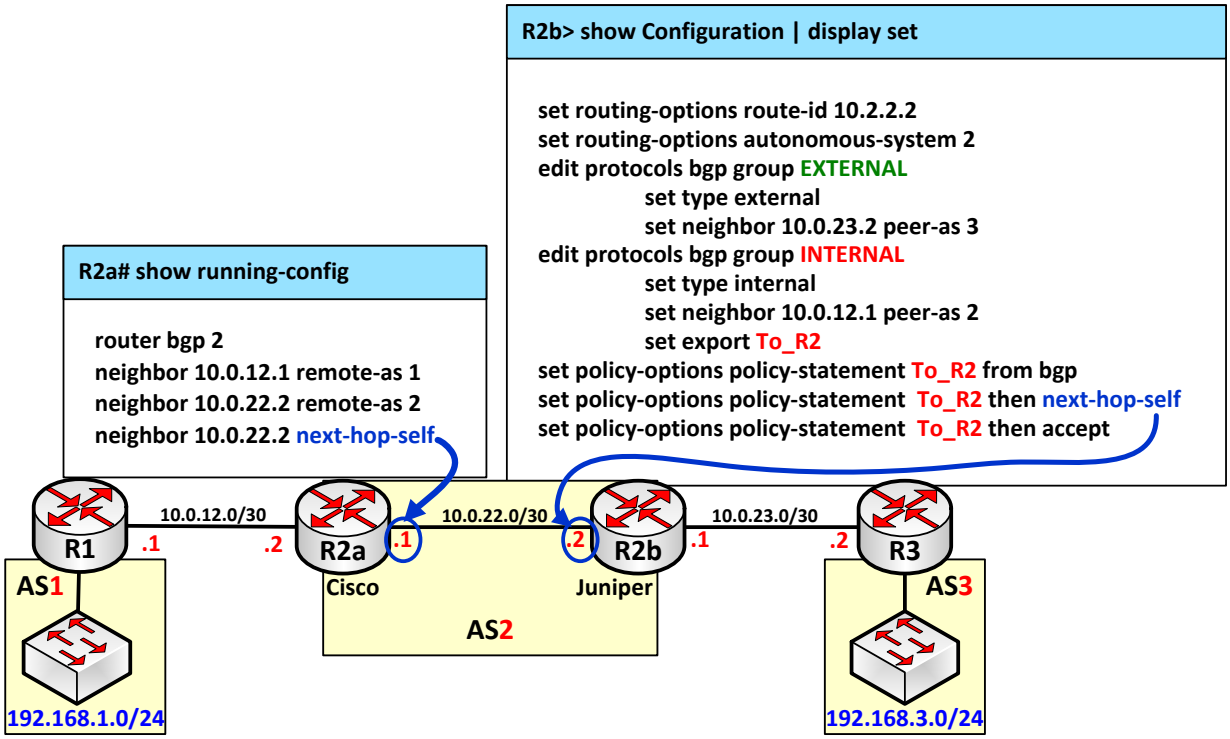


Figure 4.10

Cisco router R2a uses the command "neighbor 10.0.22.2 next-hop-self" to remove the external next hop and replace it with the interface that sources the UPDATE toward router R2b. The prefix 192.168.1.0/24 now appears with a NEXT_HOP of 10.0.22.1 instead of 10.0.12.1. Router R2b receives this UPDATE and since it knows where the NEXT_HOP 10.0.22.1 is located, it installs the prefix in its routing table. Notice that the AS_PATH correctly points to AS1.

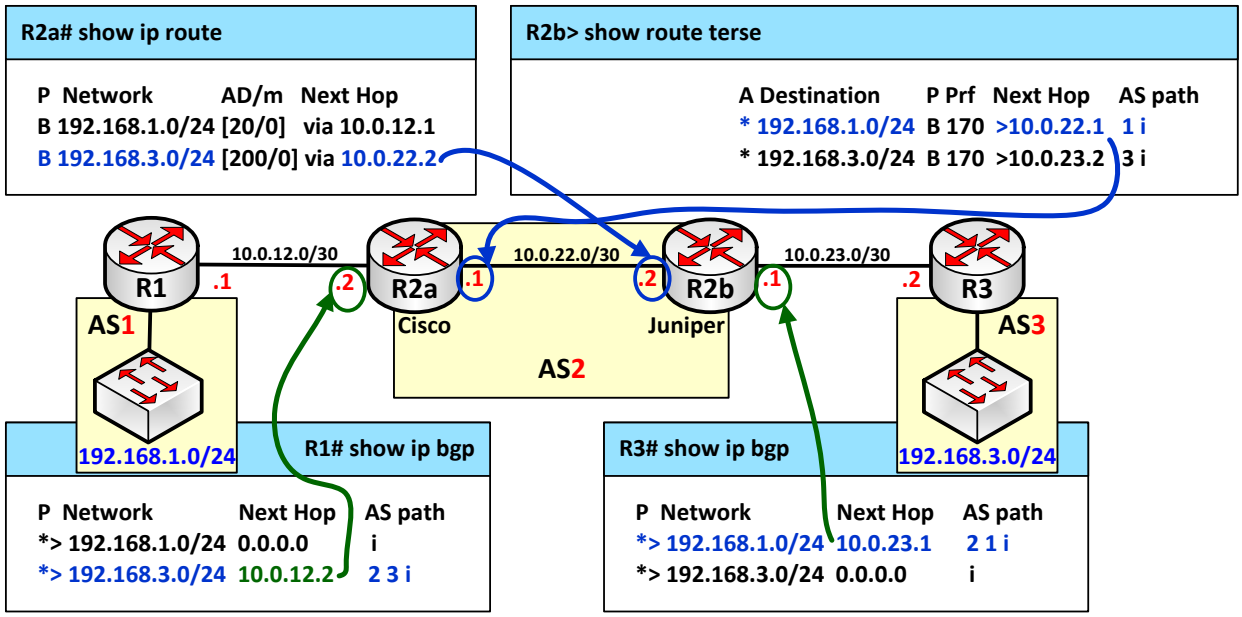


Figure 4.11

In the Juniper router R2b an export statement is created under the INTERNAL group. A small policy meaningfully named "To_R2a" matches advertisements from BGP and overwrites the next-hop attribute with self. R2b sends an UPDATE to R2a with the NEXT_HOP attribute set to 10.20.20.2. R2a receives the UPDATE, validate that the NEXT_HOP is reachable, and then it proceeds to install the prefix in the routing table.

Both R2a and R2b advertise the prefixes to their corresponding neighbours R1 and R3 and now both networks can exchange traffic. The previous basic topology has allowed the explanation of the NEXT_HOP problem; but there are more issues regarding iBGP that are explored in the next section.

iBGP inside the AS core

The network topology over this map of Southern Ontario was designed to showcase the mechanisms of **iBGP**. In real life, a network would not be designed like this if best practices were observed. (This is for educational purposes only).

AS1, located in Toronto, and AS3, located in Hamilton, need to exchange IP traffic but they do not have a physical direct connection so they use the transit services of AS2 which does have the physical infrastructure to make the connections as it is shown. More details on the physical links are not relevant to this explanation.

The external BGP peering between AS1 and AS2 and the peering between AS3 and AS2 are straightforward. AS1 in Toronto advertises the address space 192.168.1.0/24 (or network A for simplicity) while AS3 in Hamilton advertises the address space 192.168.3.0/24 (or network B for simplicity).

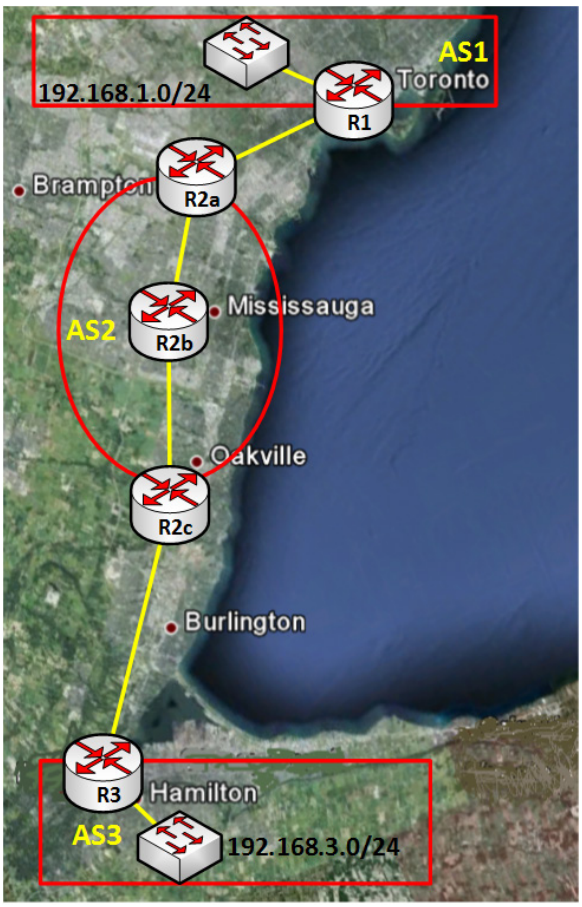


Figure 4.12

AS2 runs an IGP internally, for example, OSPF, and iBGP will be implemented gradually to expose its characteristics and issues.

Autonomous system border router R2a receives an UPDATE from R1-AS1 about 192.168.1.0/24. The route is valid and it is installed in R2a BGP table and then promoted into the IP routing table. Router R2c does the same with respect the route 192.168.3.0/24 advertised by R3-AS3. Since both ASBR have the routing knowledge, it should be just a matter of making them to talk iBGP, so they can exchange it. The configuration files for R2a and R2c are presented in the following diagram.

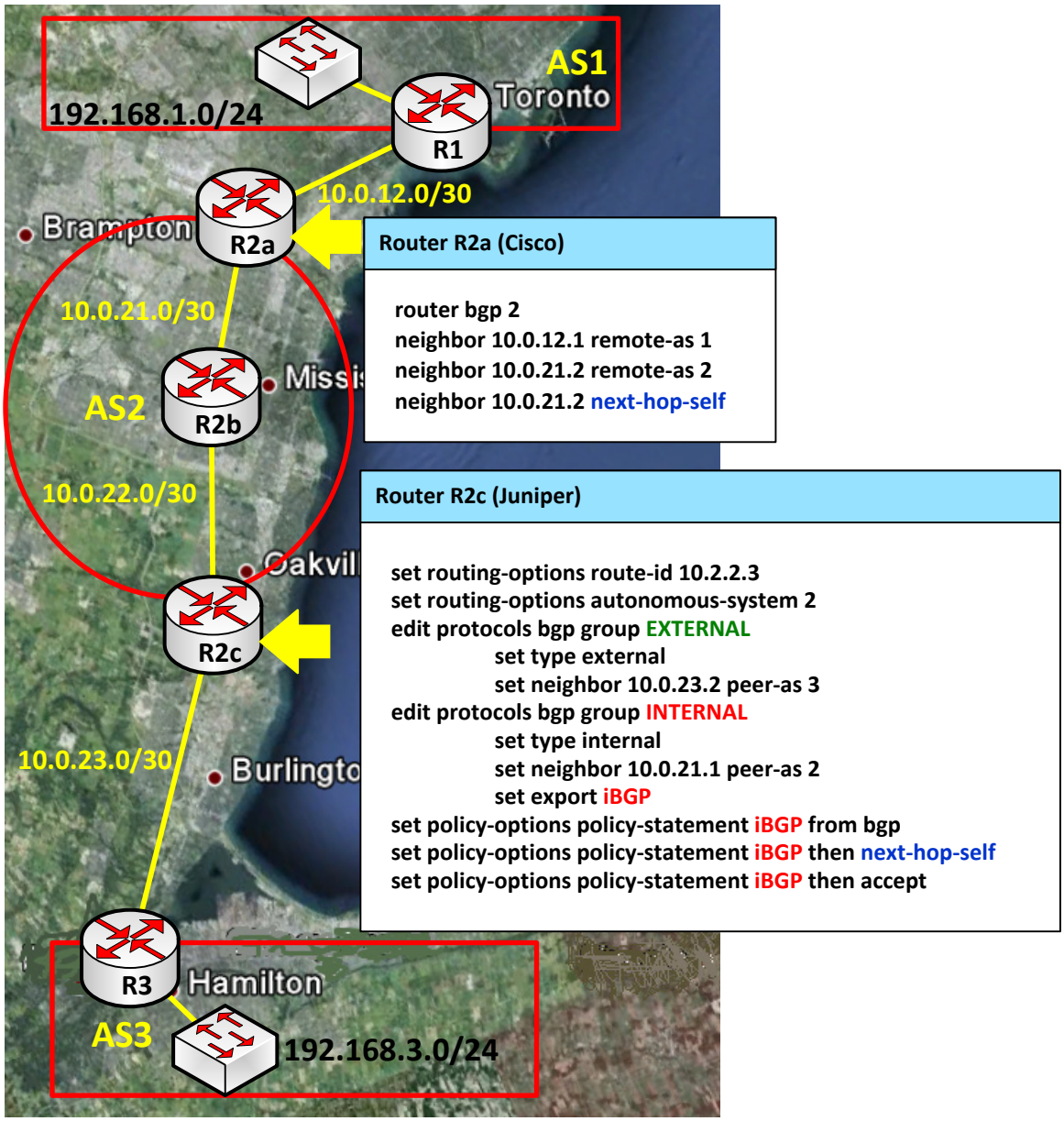


Figure 4.13

AS2 border routers are a mirror of each other. R2a is Cisco and R2c is Juniper to show the different configuration ways of both vendors. These ASBR are peering using iBGP with the NEXT_HOP set to self to make the external routes reachable. The resulting BGP tables below show that in Toronto, R1 have learned the route 192.168.3.0/24 located in AS3 (Hamilton) and R3 have learned the route 192.168.1.0/24 located in AS1 (Toronto).

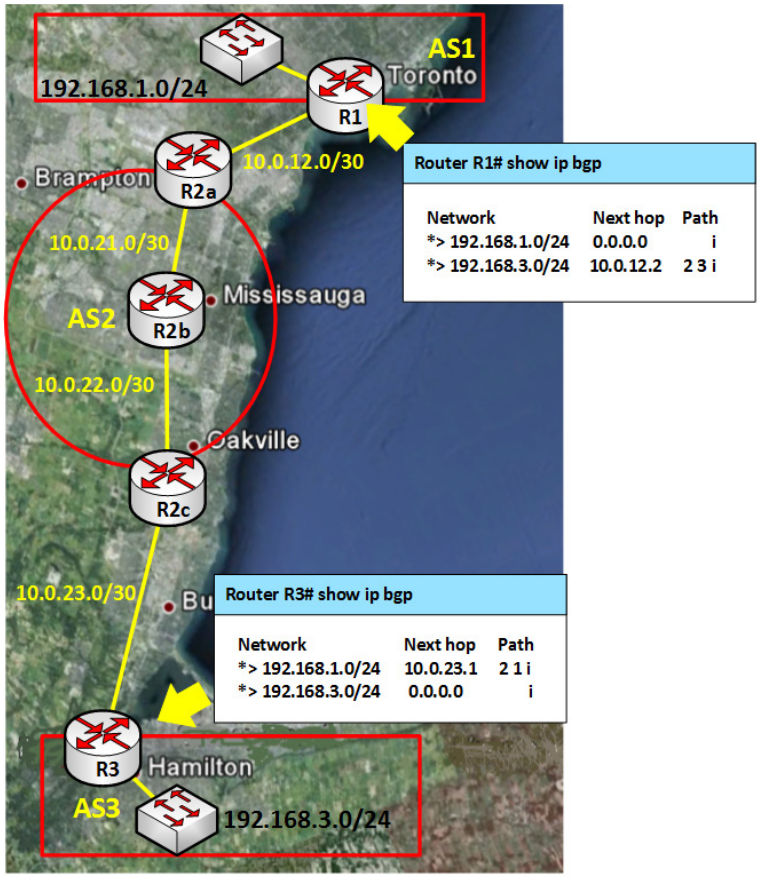


Figure 4.14

By now, IP traffic should be able to flow between AS1 and AS3 across AS2. A ping will test if this is true.

```
R1 # ping 192.168.3.1 source 192.168.1.1
host unreachable
```

The ping sourced from R1's IP address 192.168.1.1 to the IP destination 192.168.3.1 fails. Even though the end routers know the correct location of the networks, the IP traffic cannot be delivered. It is time to do some troubleshooting to find the cause. A combination of traceroute, show commands and sniffing helps to gather the information portrayed next:

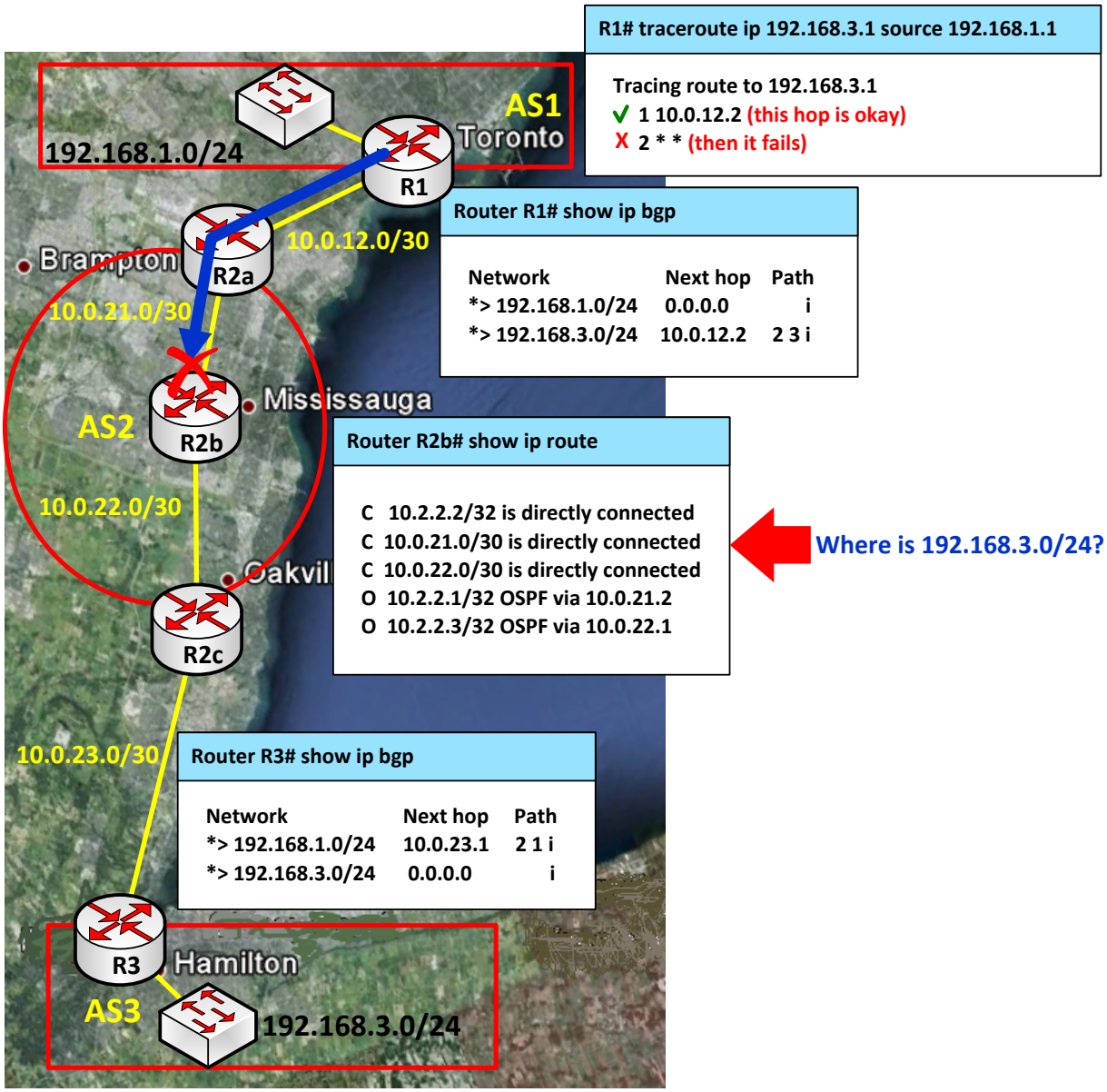


Figure 4.15

The IP traffic is dropped in R2b inside the core of AS2. The show ip route command issued in R2b shows that this router IP knowledge is limited to the prefixes inside AS2, so it has no idea where the external routes are located. Basically, no one ever told the router anything since it does not speak BGP with the border routers. Router R2a knows that the destination 192.168.3.1 is advertised by R2c, so it sends the IP packets to the only possible way which has R2b in its path. When the IP packet arrives to R2b, this router looks up its routing table, it does not find a match, and then, it drops the packet. The network looks fine from the viewpoint of all the routers except for R2b.

The original BGP implementations used to "synchronize" the Internal Gateway Protocol IGP with BGP. This was called the **Synchronization Rule**: "the external routes should be present in the IGP before they can be used by BGP". Currently, all vendors disable the synchronization rule by default since importing the Internet routing table into the IGP is a terrible idea. IGPs are not designed to handle a large number of routes like the current (as for July 2017) Internet routing table of more than 645,000 IPv4 routes.

To fix this problem and to avoid redistribution of routes into the IGP, AS2's border routers are going to be reconfigured to talk iBGP to R2b next. R2a peers now with R1 using BGP and with R2b and R2c using iBGP. This is R2a BGP configuration snippet.

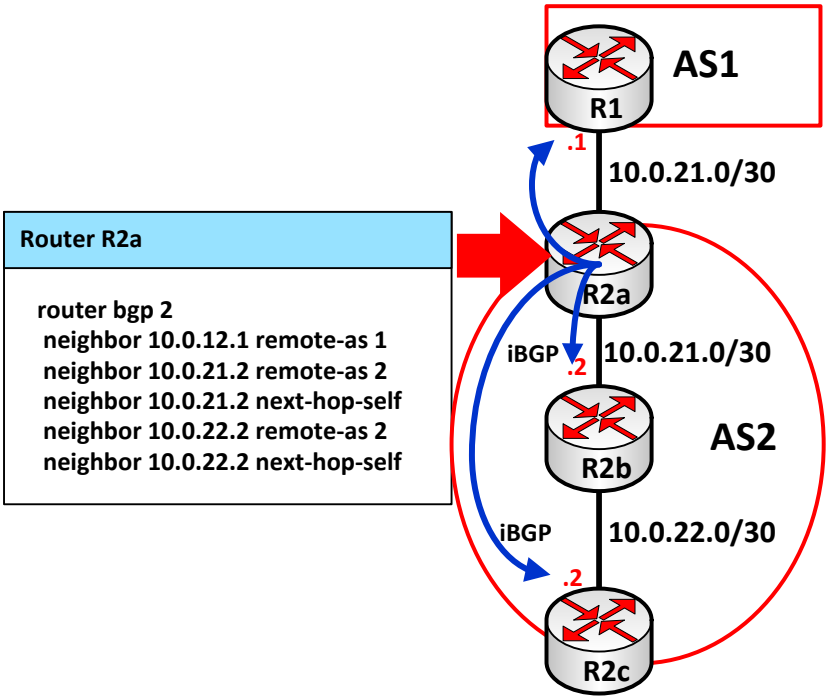


Figure 4.16

R2b peers, using iBGP, with both of the ASBRs. There is no need to do NEXT-HOP-SELF because the router is completely inside the core of the AS.

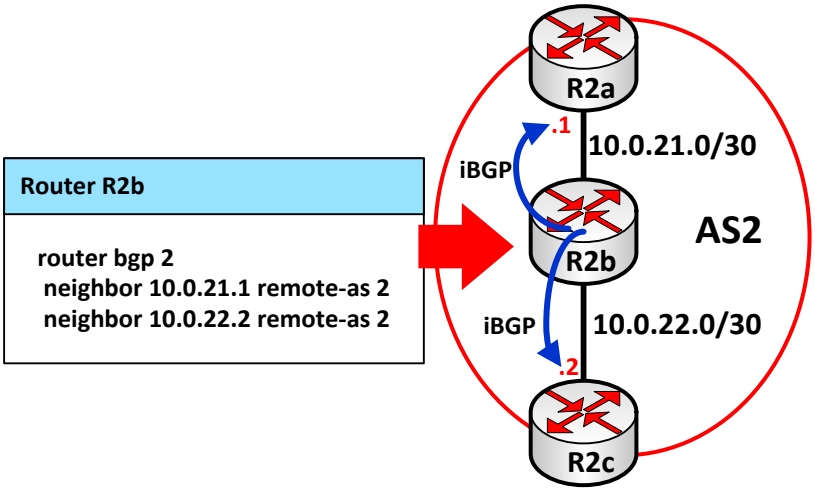


Figure 4.17

Finally, router R2c peers with R2a and R2b and this is its configuration snippet:

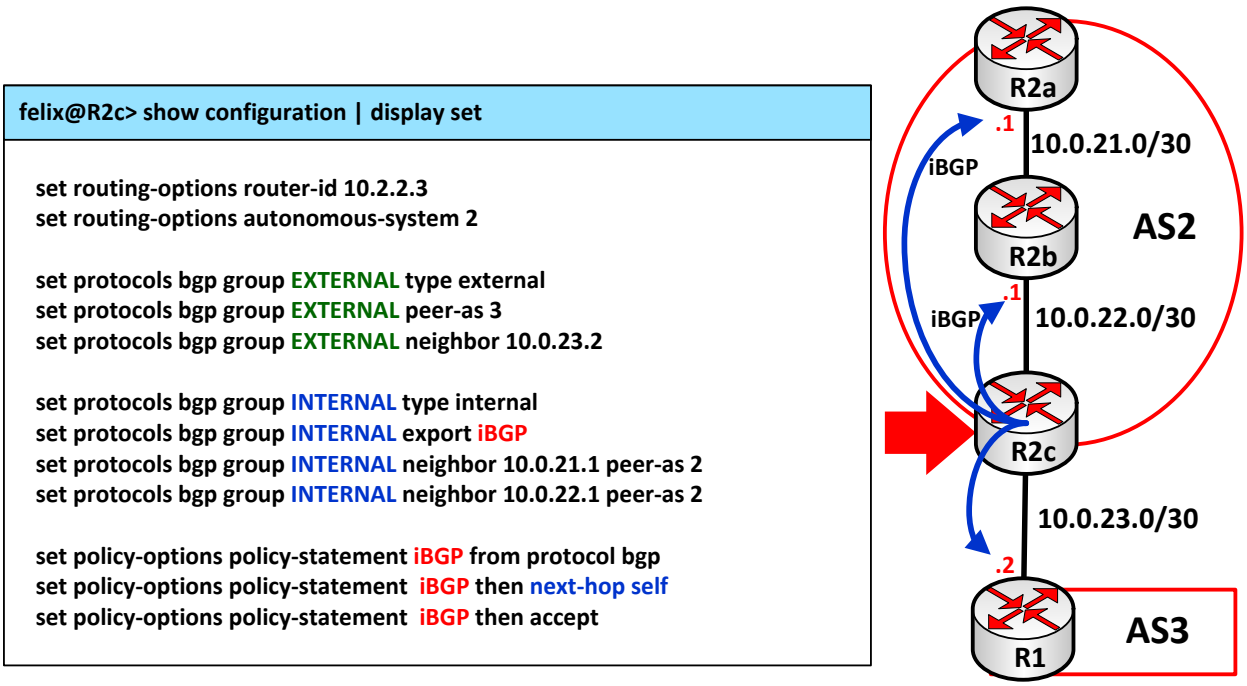


Figure 4.18

The result of this configuration is observed in Router R2c:

```
felix@R2c> show route protocol bgp terse
A Destination        Protocol   Preference   Next hop      AS path
* 192.168.1.0/24       BGP        170        >10.0.21.1      1 i
* 192.168.3.0/24       BGP        170        >10.0.23.2      3 i
```

The IP route 192.168.1.0/24, located in AS1, is reachable via 10.0.21.1 so everything seems to be in place. A ping or a traceroute from R1 to R3 should verify the reachability.

```
R1#ping 192.168.3.1 source 192.168.1.1
Success rate is 100 percent

R1#traceroute 192.168.3.1 source 192.168.1.1
 1  10.0.12.2
 2  10.0.21.2
 3  10.0.22.2
 4  10.0.23.2
```

The configuration ended up in a BGP full mesh logical peering, even though there is no physical connection between R2a and R2c, a TCP session keeps BGP established between these routers.

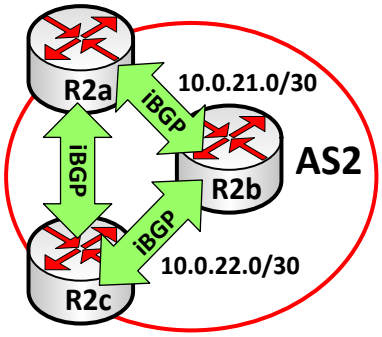


Figure 4.19

At this point the administrator could reason that the BGP peering between R2a and R2c is not required and it proceeds to remove it. The "logic" would be that since there is only one path anyway there is no point in maintaining another BGP session. Consequently, the BGP peering between R2a and R2c is removed and these are the results:

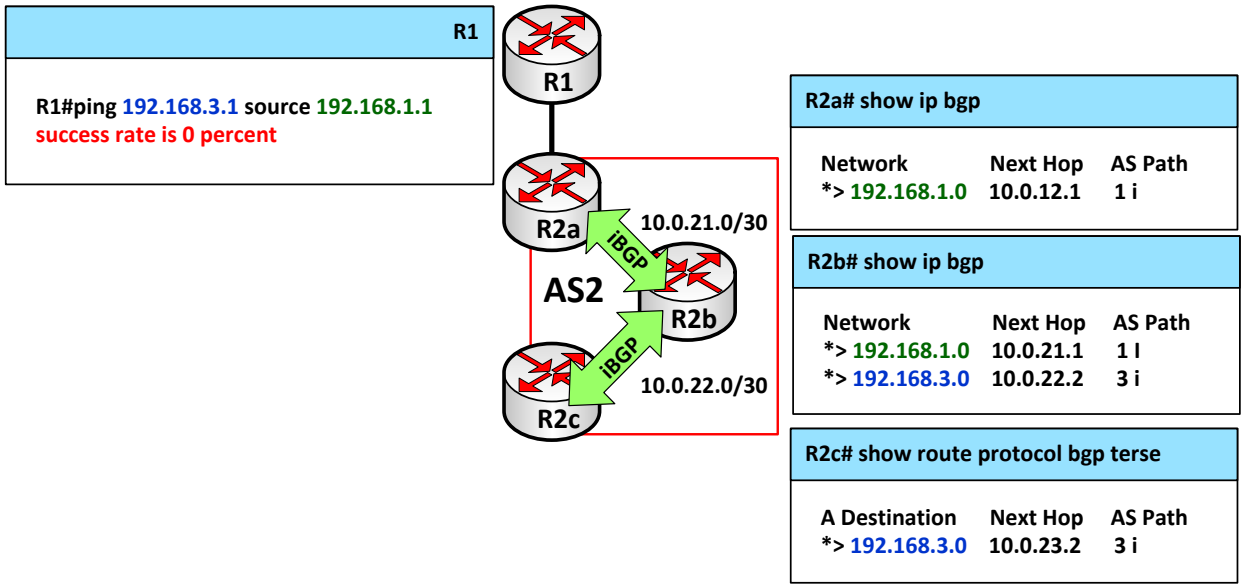


Figure 4.20

The networks have become unreachable again. Routers R2a and R2c have lost knowledge of the networks across AS2. However, router R2b stills knows both IP routes 192.168.1.0/24 and 192.168.3.0/24. For some reason, R2b is not propagating the routing knowledge. To find out the reason of the problem, the BGP session between R2a and R2b is restarted while a sniffer captures the BGP traffic. (**Note**: in a real life production operation, clearing commands must be avoided or performed with great care and with the proper authorization.)

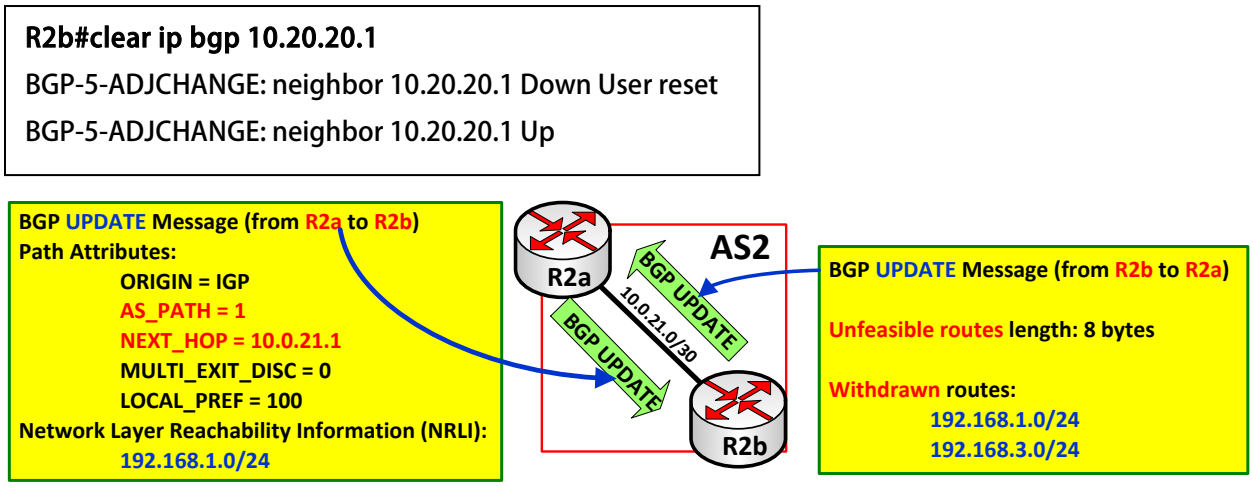


Figure 4.21

The clear command forces the peer adjacency R2b-R2a to go down and then back up. This causes the exchange of UPDATES that is shown above. R2a advertises 192.168.1.0/24 to R2b as

it should, but then R2b reacts with an UPDATE to withdraw 192.168.1.0/24 and 192.168.3.0/24. Basically, R2b is telling R2a: "you cannot reach these networks thru me". Moreover, R2b does the same with respect R2c. It is evident that R2b obtains the knowledge from the ASBRs, but why is R2b sending withdrawal UPDATES to its neighbours?

There is a rationale for this behaviour and it has to do with BGP fundamentals. Let's observe the sniffed BGP UPDATE from R2a to R2b regarding the AS_PATH attribute. The border router R2a did not attach AS2 into the AS_PATH string which is still AS1. There must be a reason.

The major defense that BGP has against routing loops is the AS_PATH attribute. When a BGP router sees its own AS number in the AS_PATH string, it stops the propagation of such route advertisement. That is why the border router R2a cannot add AS2 to the AS_PATH string; otherwise, the other routers speaking iBGP in the same AS would reject the UPDATEs. That is actually a BGP rule:

Attaching an AS number to the AS_PATH rule

- A BGP speaking router can only add its autonomous system number (ASN) when the UPDATE is leaving toward a different autonomous system.

In short, only ASBR can add AS numbers to the AS_PATH and routers know that because they compare their own AS number with the destination AS number.

However, this bring about a dilemma, the application of this very same rule would leave the iBGP routers vulnerable to routing loops. The following sequence of diagrams illustrates the typical case of a routing loop that might appear in distance vector protocols. In principle, iBGP would be prone to this type of problem if there were no countermeasures.

For example, an unfortunate sequence of events could cause a router to be misled about the location of network A. Initially, R1 receives, installs, and propagates the right information to R2 and R3. But soon after, in step 2, network A fails and another UPDATE withdraws the prefix. R1 propagates the withdrawing information to both R2 and R3. Unluckily, in a matter of bad timing, the withdrawing UPDATE bound to R2 gets delayed while R3 does get the UPDATE and removes the route.

In step 3, R2 had a queued UPDATE advertising network A and so it sends it before it can get the withdrawing UPDATE from R1. Since all the routers are doing "next-hop-self" then R3 learns

that network A appears to be reachable via R2. Finally in step 4, R3 advertises this "mirage" of network A to R1 looking like it is reachable via R3.

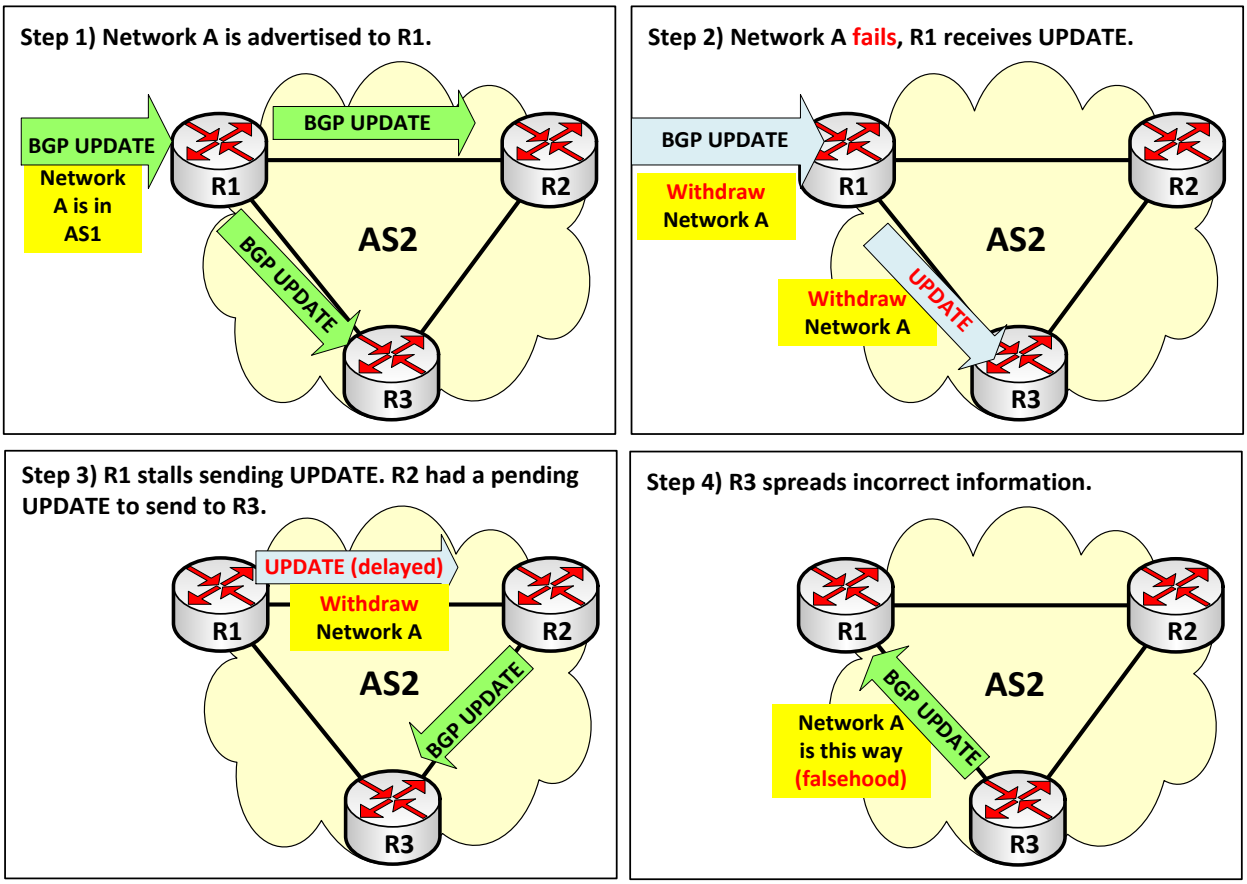


Figure 4.22

As a historical example of this type of problem, the original distance vector protocol RIP used to suffer from this type of routing loop and it took a deal of redesigning to add countermeasures such as counting to infinity, split horizon with poison reverse, and triggered updates to mitigate the problem. Nevertheless, RIP has a metric on which to implement these remedies while BGP does not have an internal metric because BGP is simply not an IGP. Then how can iBGP be protected from this kind of routing loop? The solution is another BGP rule:

One-hop iBGP rule

- An iBGP speaking router cannot further propagate routing information that has been learned via iBGP.

An ASBR can advertise an external route to an iBGP router, but such iBGP router cannot tell another BGP router about the route; therefore, the external routing information only travels

one hop count inside an AS. This rule solves the problem presented in the previous diagram. For example, R2 could not have advertised network A to R3 if the one-hop iBGP rule had been in place.

Going back to the original network, this explains why R2b learned prefixes 192.168.1.0/24 and 192.168.3.0/24 but it would not propagate the information to the ASBRs R2a and R2b. Even more, R2b reacted with UPDATEs withdrawing the routes. That is yet another BGP rule:

iBGP split horizon with poison reverse rule

- An iBGP router will send back to an ASBR, an UPDATE withdrawing the routes that have been advertised by such ASBR.

This is a way of telling the ASBR "those networks are unreachable this way."

iBGP in Full Mesh Configuration in the Core

It is clear now that the example topology has functional issues because the logical session is established over a single path, basically an improper physical layout. To fix this problem, a physical connection between R2a and R2c is added. An iBGP session must also be established over such link. Now every router is compliant with the one-hop iBGP rule.

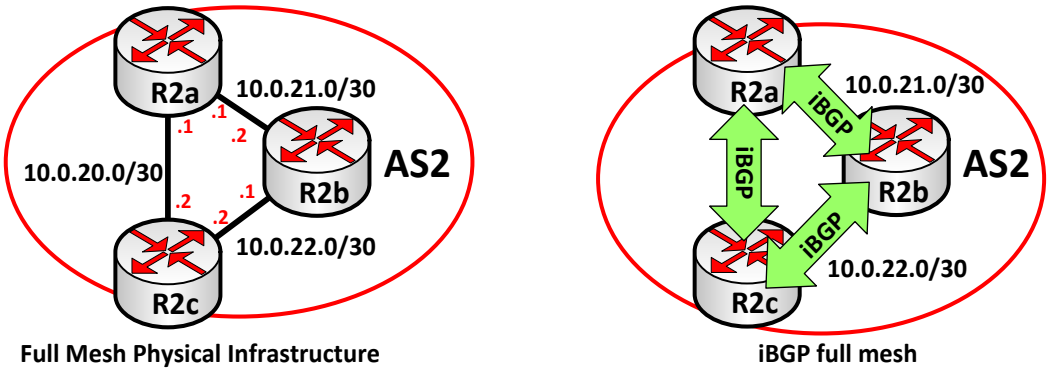


Figure 4.23

The common practice is that the peering relationships are established using the loopback interface addresses of the routers which are also the BGP router-id.

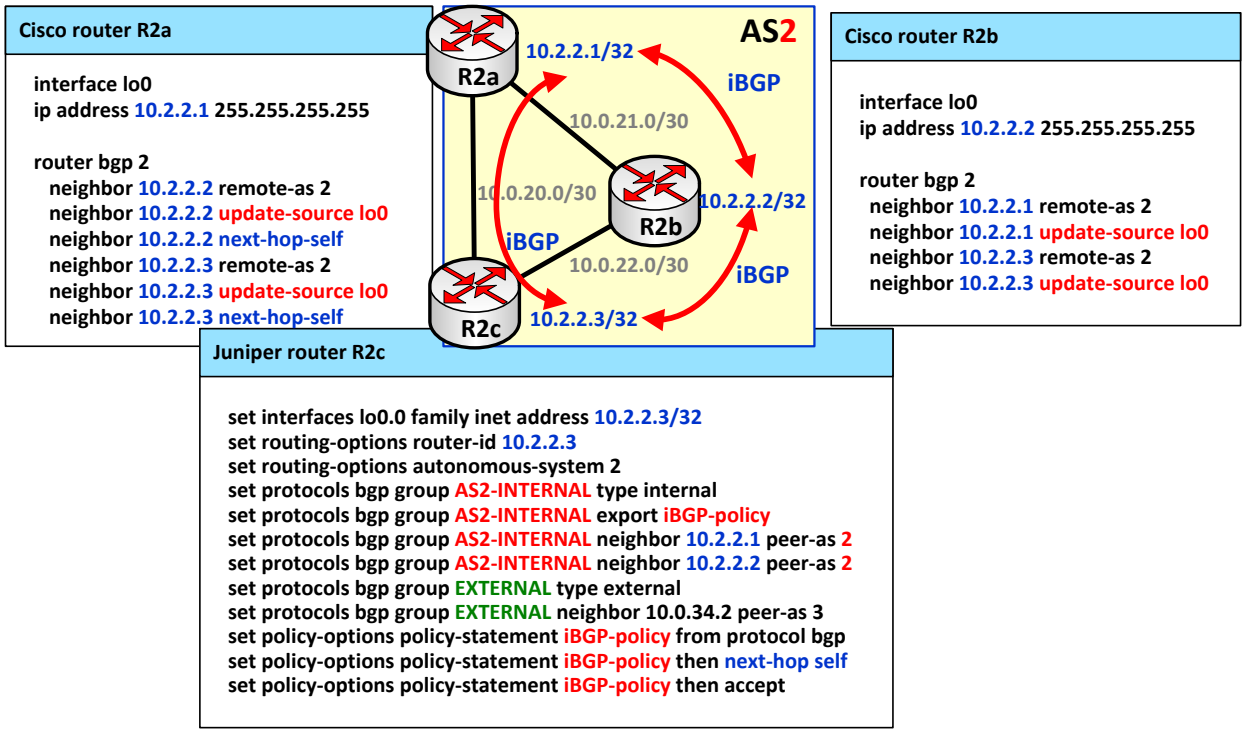


Figure 4.24

Loopback interfaces are software based so they are always up and as long as the IGP advertise their presence, the BGP session will be up. For instance, if the interface 10.0.22.2 fails, there is still a valid path to reach the loopback address 10.2.2.3 via 10.0.20.2. In this case, the TCP session that transport BGP stays up.

The Cisco IOS used in this example requires that the "neighbor" command be accompanied with another command with the "update-source lo0" like this:

```
router bgp 2
  neighbor 10.2.2.2 remote-as 2
  neighbor 10.2.2.2 update-source lo0
  neighbor 10.2.2.2 next-hop-self
```

The command is essentially a fix to the IOS software. Normally, a Cisco IOS router is expecting the BGP OPEN message to be sourced from an address in a common DMZ link. However, the configuration file is telling the router to open the session to a loopback address. In the figure below, router R2a sends the IP packet that contains the BGP OPEN from source address 10.0.21.1 to the destination 10.2.2.2. On the other side, R2b rejects the BGP opening with a TCP FIN that terminates the conversation.

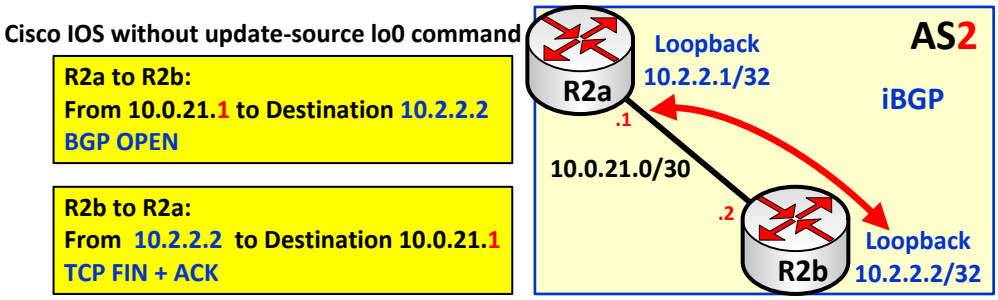


Figure 4.25

The command update-source lo0 enables this type of connections as this snippet proves it:

```
R2a(config-router)#router bgp 2
R2a(config-router)#neighbor 10.2.2.2 update-source lo0
        %BGP-5-ADJACENCY: neighbor 10.2.2.2 Up
```

The one-hop rule forces network designers to build full mesh topologies but sometimes, it is not feasible to install the full mesh physical infrastructure because of the costs or the distance. In this case, there is a BGP feature that allows a controlled override of the one-hop iBGP rule.

iBGP Route Reflector

A route reflector is an iBGP speaking router located strategically inside the AS core in a position that can reach other iBGP routers. Basically, a route reflector acts as a relay of external routing information bypassing the one-hop iBGP rule. In the figure below, only R2b can be a route reflector.

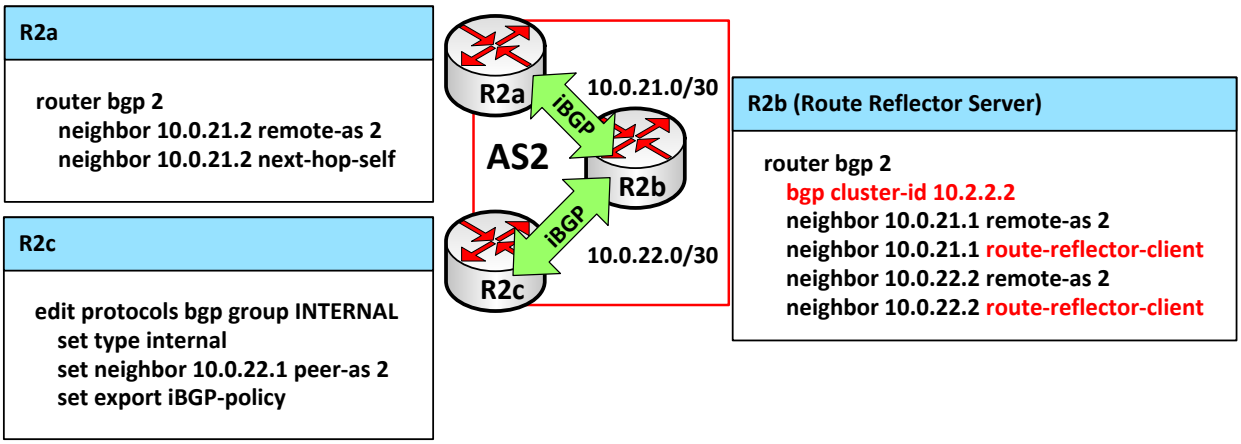


Figure 4.26

The Cisco configuration is very simple; in R2b, the route reflector clients are listed by IP addresses. The command BGP cluster-id 10.2.2.2 sets the route reflector identification.

In JUNOS, the configuration of a route reflector in JUNOS is just this command:

edit protocols bgp group EXAMPLE **set cluster** (Router-id IP address goes	→	**edit protocols bgp group EXAMPLE** **set cluster** 10.2.2.2

The behaviour of the route reflector can be analyzed using the sniffed conversation between the routers in AS2 which is displayed in the diagrams that follow. ASBR R2a learns 192.168.1.0/24 (network A) from an external BGP session so it proceeds to UPDATE its internal BGP neighbour R2b with a next-hop set to itself.

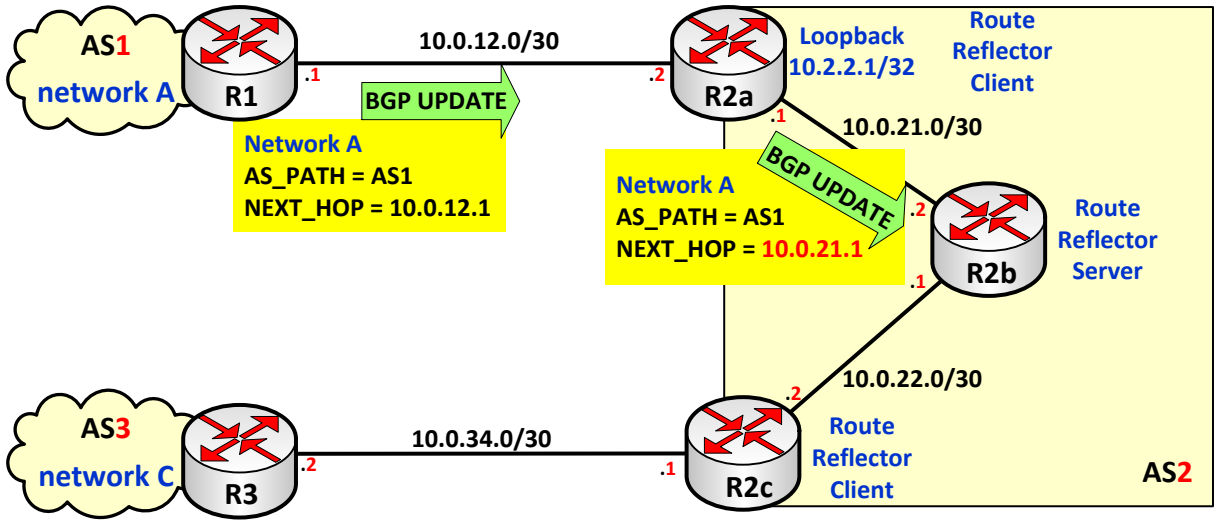


Figure 4.27

Router R2b is the **Route Reflector Server** and it relays the external information to its client R2c. It does so by keeping the NEXT_HOP unchanged and adding two new BGP attribute, the **ORIGINATOR** and the **CLUSTER-ID**.

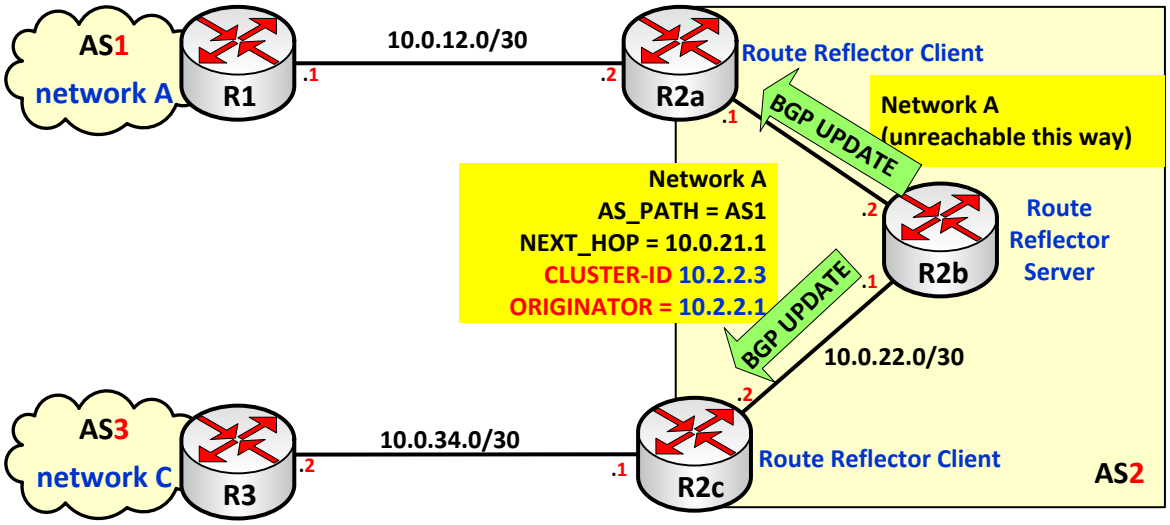
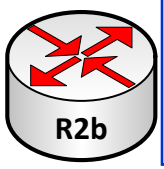


Figure 4.28

Additionally, the Route Reflector Server sends back to R2a and UPDATE with a withdraw message for the same prefix. This is an application of the technique of **split horizon with poison reverse** used in distance vector protocols for protection against routing loops. In short, R2b tells R2a: "you cannot reach 192.168.1.0/24 coming this way."

When a Router Reflector Server relays information to a client, it attaches two attributes that pertain to the route reflection function. This UPDATE from R2a to R2c shows the details.



Border Gateway Protocol
Marker FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
Length: 21 Bytes
TYPE: (number 2) UPDATE Message
Path attributes:
ORIGIN = IGP
AS_PATH = 1
NEXT_HOP = 10.0.21.1
MULTI-EXIT-DISCRIMINATOR = 0
LOCAL-PREFERENCE = 100
CLUSTER_LIST = 10.2.2.3
ORIGINATOR = 10.2.2.1
Network Layer Reachability Information NLRI
192.168.1.0/24

Figure 4.29

- The NEXT_HOP remains unchanged to the gateway that introduced the prefix into the autonomous system.
- The CLUSTER_LIST is set to the router-id of the route reflector server. An autonomous system could have more than one route reflector server, in that case, the CLUSTER_LIST identifies what RR server is generating the UPDATE.
- The ORIGINATOR indicates the identification of the AS border router that introduced the prefix into the autonomous system.

Both attributes, the **CLUSTER_LIST** and the **ORIGINATOR,** are optional, non-transitive and complete. They are optional because they only appear when there is a route reflection. They are non-**transitive** because they only have meaning inside the AS. Finally, they are complete since all the routers understand their meaning.

The NEXT_HOP is kept unchanged because the routers in this routing domain are supposed to know all the internal connections advertised by an IGP which in this example case is OSPF. Also, routers in the core would use this information to calculate a shortest path to the AS exit point if the same route is introduced by more than one border router.

The **CLUSTER_LIST** (attribute type 10) contains the router-id of the router that reflects the routing information. A route reflector with its clients conform a **Route Reflection Cluster**[1]. The CLUSTER_LIST is specially needed when there is more than one route reflector in the AS. For example, one AS that has three RRs would need three cluster identifications to avoid routing loops. In certain topologies, a route that has been reflected could, in principle come back to the original route reflector. If there is more than one RR, the CLUSTER_LIST contains the list of the route reflectors that have relayed the route information. The CLUSTER_LIST attribute is examined by the router reflectors and if they see their own router-id, the route is not reflected again.

In a large AS, the cluster can get more complicated and it is possible to have more than one RR per cluster. This is to provide redundancy if one of the RR fails, the others can answer. This type of topology has to be carefully designed. In this case, all the route reflectors use the same CLUSTER_ID which is a 4 byte number, typically an IP address.

The **ORIGINATOR_ID** (attribute type 9) is the IP address of the router that introduces the routing information into the autonomous system. For example, router R2c uses the IP address 10.2.2.3 as its router-id. Then R2c UPDATES the route reflector R2b with external information

from AS3. When R2b reflects the route, it attaches the router-id of the originating router R2c. This information is very helpful for the routers to determine the shortest exit point toward the destination.

Core Free of iBGP

The configuration and maintenance of iBGP in a large routing domain can be challenging. The deployment of clusters of route reflection requires plenty of planning to get it right and in many opportunities, the failure of a device triggers a whole set of unforeseen problems. An alternative to the implementation of iBGP inside an AS is not to have it at all.

The following diagram will illustrate the idea. Autonomous system AS10 has four border routers (ASBR R2, R5, R8, and R11) that speak BGP. The rest of the routers (R3, R4, R9, and R10) do not speak BGP at all. Furthermore, an IGP is running inside the AS and all internal prefixes are reachable by all routers. Each router has a loopback address and that is critical for the functioning of this model because the border routers (R2, R5, R8, and R11) are iBGP peering with the loopback addresses in a full mesh as indicated by the red lines in the diagram. The paths followed by the BGP messages are not relevant as long as there is any path to reach the loopback on each router.

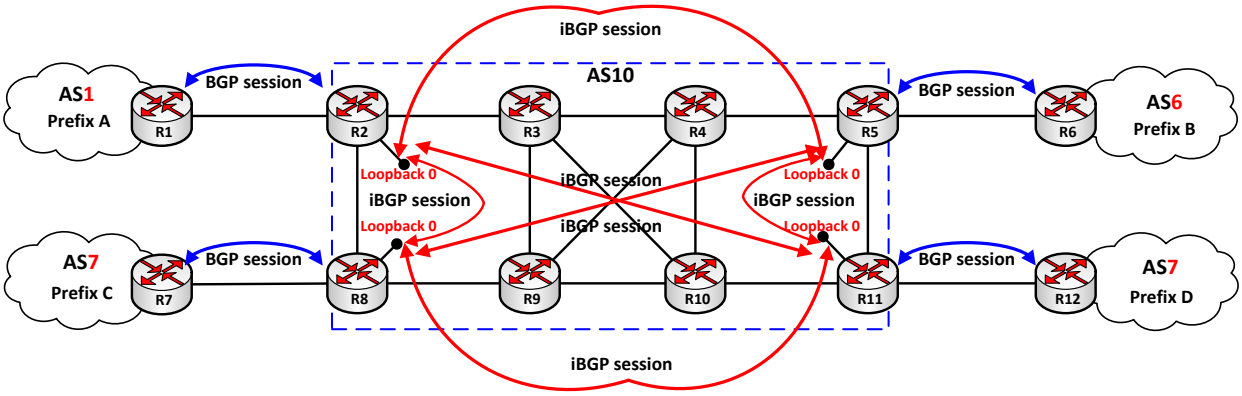


Figure 4.30

Routers R1, R6, R7 and R12 peer with the corresponding neighbours in AS10 with just regular BGP. For instance, when R1 advertises prefix A to R2, this border router will re-advertise the prefix to R8, R5, and R11 with the next-hop-self set to its own loopback address. The same situation happens in all the edge routers of AS10. All ASBR get to know the external prefixes and they all implement next-hop-self internally.

Since R5 knows that prefix A is located in AS1 and its next hop is the loopback address of R2, then it installs that prefix in its IP routing table and then it proceeds to advertise the prefix to its external neighbour R6 in AS6. The same principle apply everywhere, so the knowledge about all routes propagates effectively.

However, this configuration as it is, presents a crucial problem which is better explained with an example. When an IP packet starts flowing from a source in network A to destination in network B, R2 knows that it has to send it to the advertising router R5. In consequence, R2 delivers the IP packet to the next router in the path. Router R3 receives the packet, then it looks up the destination IP address of route B, but since it does not speak BGP, it has no clue about it and it just drops the packet. All the core routers would drop the IP transit traffic between the external routes because they lack the knowledge about them. Basically, it is a useless configuration if it is left as it is right now.

However, if only the internal routers could deliver the data without doing an IP look up on the packets, then this model would work. That is actually possible with the implementation of encapsulation tunnels. There are several ways to implement tunnels, but the basic idea is the same as the next diagram explains.

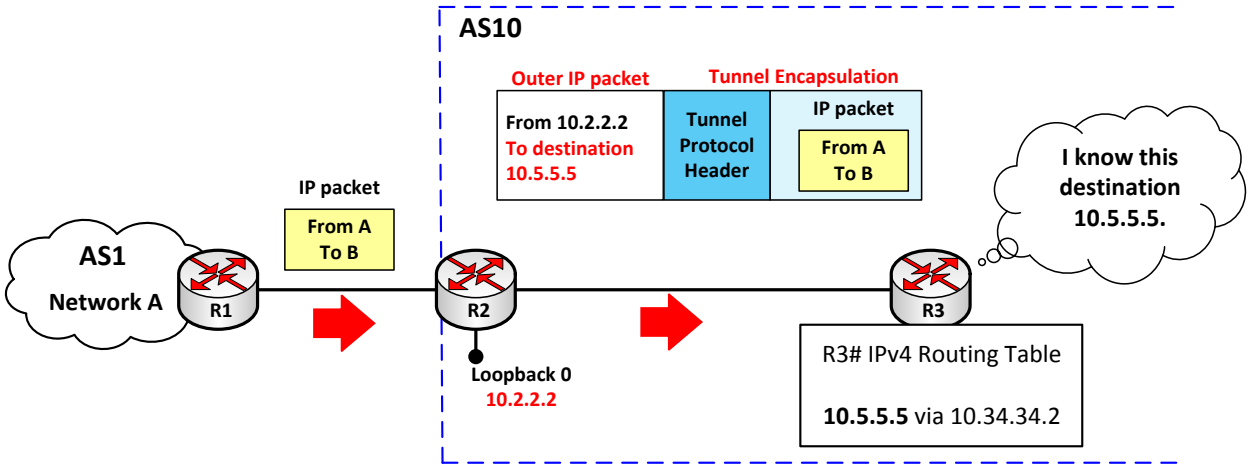


Figure 4.31

Previously, a type of tunnel has been configured between R2 and R5. When R1 sends IP traffic toward destination B, the edge router takes the IP packets and place them inside the payload of a tunnel protocol and then it places everything inside the payload of an IP packet that has as destination address, the loopback address of R5. Then it sends the IP traffic via the next router which is R3. This router looks up the destination address 10.5.5.5 in its routing table and then forwards the packet toward R4. The same process is repeated in R4 until the packet is received

by the tunnel destination. Finally, R5 removes all the outer encapsulation and that reveals the payload which is the initial IP packet sent by R1. Since R5 knows where destination B is located, it just forwards the IP packet toward R6 in AS6.

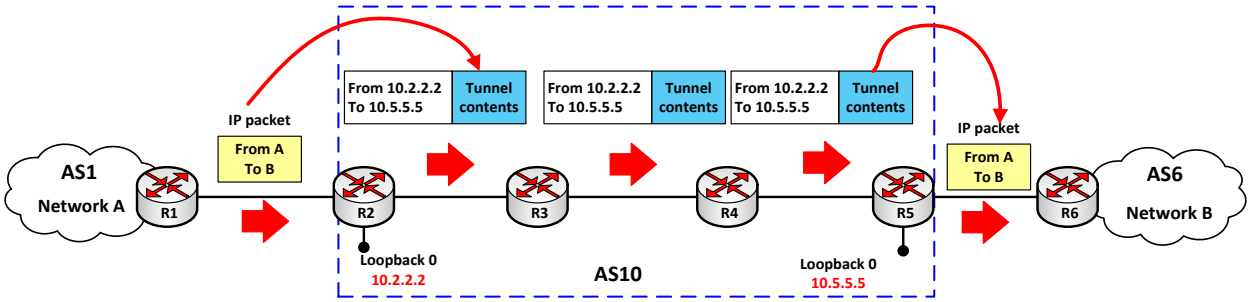


Figure 4.32

Tunnel encapsulation basically allows the autonomous system to deliver traffic transparently from edge to edge without having to introduce BGP routing information inside the core. This has additional benefits. First, the configuration of the core routers is simpler and second these routers have very lean routing tables for the fast forwarding of packets.

One way to implement the previous model would be the GRE protocol. But, this protocol requires the manual configuration of the tunnels between all the edge routers. Instead, a dynamic solution is preferred and that is the Multi-Protocol Label Switching MPLS. This is a switching technology that encapsulates IP, or any kind of traffic, inside the payload of the MPLS protocol packets. This is the basic idea of an MPLS model delivering IPv4 traffic from A to B.

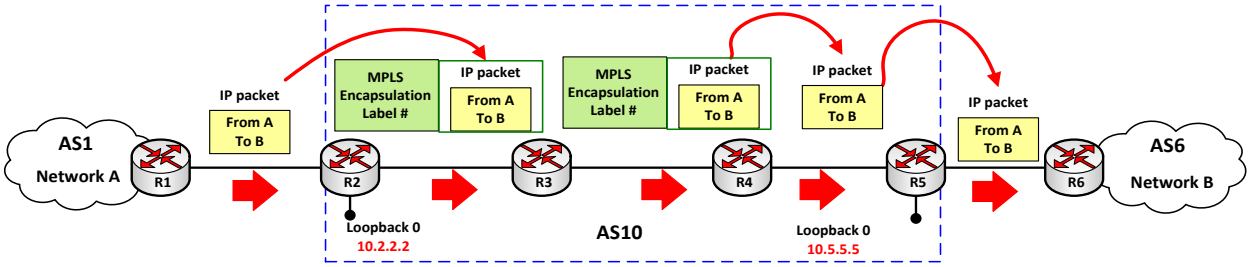


Figure 4.33

The diagram shows, in a simplified way, the transit of an IPv4 packet from network A located in AS1 to network B in AS6. The edge router R2 encapsulates the IPv4 packet inside the payload of an MPLS packet and then encapsulates that inside a layer 2 (Ethernet) frame (not shown in the diagram to simplify the explanation). The MPLS header has a label that points to the loopback address 10.5.5.5 of R5. The frame is delivered to the next router R3 which proceeds to read the label number in the MPLS header. The switching of the packets, inside AS10, is entirely

done by looking up the MPLS labels instead of the IP addresses and these labels have been previously negotiated by the routers using a protocol for label distribution. The labels pertain only to the prefixes inside AS10. Basically there is a label per each internal prefix.

The following router R4 also performs a lookup up of the MPLS label and since it is directly connected to the destination edge router, it just removes the MPLS encapsulation and delivers the IPv4 packet within an Ethernet frame to R5. This router speaks BGP so it knows of what to do with this IP packet. R5 looks up its routing table and then forwards the packet to AS6.

Routing domains that implement MPLS, and all large carriers and ISPs do, have removed iBGP from the core and they have only left the AS border routers to talk iBGP among themselves. That is the standard BGP-free core solution and these are its fundamental elements:

- Only the Autonomous System Border Routers ASBRs need to peer using iBGP.
- The iBGP peering is done using the loopback addresses of the routers. Since the loopback addresses are advertised by the IGP, as long as there is any physical path, the routers are reachable.
- If all the ASBR are peering in a full mesh logical topology there is no need for route reflection. Alternatively, route reflectors can be strategically deployed in the core so full mesh logical topology is not a requirement.
- The AS core runs the protocol Multi-Protocol Label Switching MPLS.
- MPLS establishes logical paths where the core routers switch MPLS packets that contain the IP packets. Fundamentally, MPLS encapsulates the IP traffic from edge router to edge router.

The cylinders of different colours represent MPLS labels in the following network. It is easier to explain with colours, but in reality MPLS uses numbers as labels. Look at this as a pipeline with sections of different colours where traffic can be encapsulated. To reach any destination inside this network, each router only needs to encapsulate the IP traffic inside MPLS packets with labels that point to the destination. For example, to router R2, an incoming yellow label represents a destination toward edge router R4, so R2 swaps the label to green and switches the MPLS packet toward R3. This router receives the MPLS packet with green label and decides that R4 only needs an IP packet. Since R4 is a border router with external BGP knowledge, it will know what to do with the traffic.

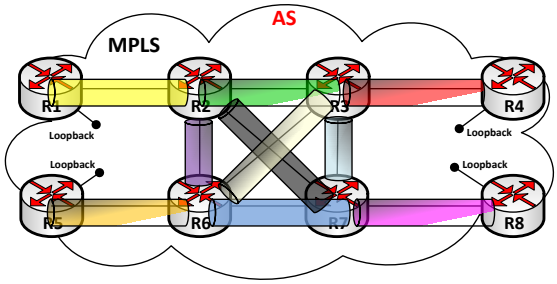


Figure 4.34

An MPLS core free of iBGP offers an elegant and simple solution to the issues of maintaining iBGP inside a core. Even better, to avoid having a full iBGP mesh, some internal routers can be made route reflectors to manage regional sections of the network like in this diagram.

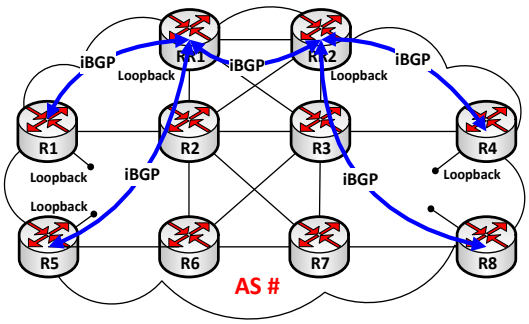


Figure 4.35

RR1 manages a route reflection cluster with R1 and R5 as clients while RR2 manages its cluster with R4 and R8 as clients. Also, the RRs reflect to each other. In a large AS with many edge routers reducing the number of BGP adjacencies is a necessity and this RR model achieves that. BGP has an essential role in the implementation of the MPLS service model. For this reason, a whole chapter in this text is dedicated to it.

Chapter Summary

BGP is used not just to advertise routing knowledge among different autonomous systems, but also to continue the propagation of such knowledge throughout the AS. When BGP is spoken inside an AS, it is called internal BGP or iBGP. This is the same protocol but some features and rules tweaked to allow routers to speak BGP in the same autonomous system.

The configuration of iBGP requires that the network engineer understands the inner mechanism of the protocol since modification to the default behaviour of BGP is required. Specifically, a thorough understanding of attributes is needed, such as NEXT_HOP, AS_PATH and additional attributes that appear in route reflection such as ORIGINATOR and CLUSTER_LIST.

Exercises

These exercises are designed to enhance your understanding of the mechanisms and issues of iBGP.

Exercise Route Reflection

This is a typical topology to demonstrate route reflection issues.

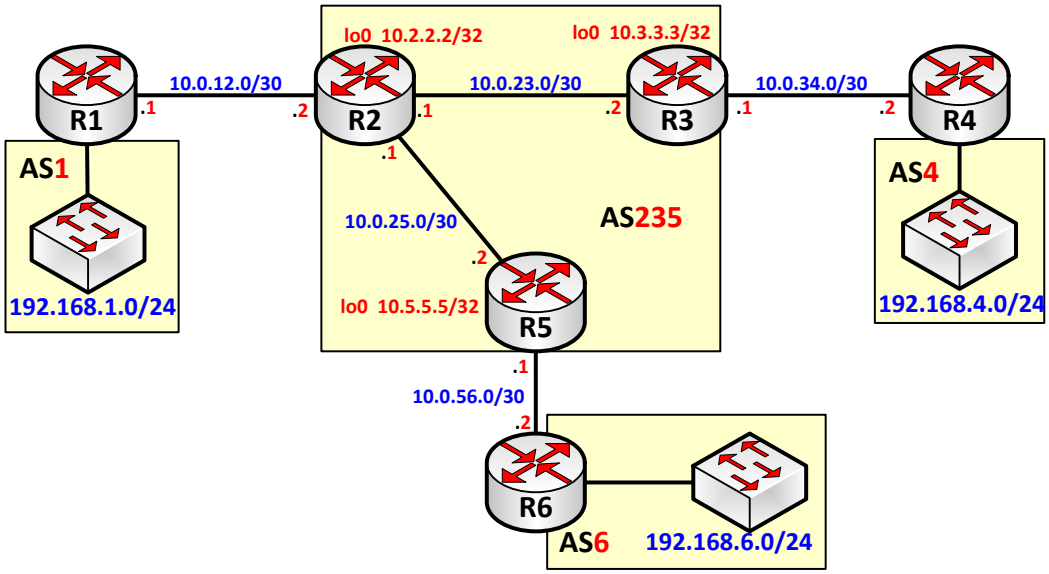


Figure 4.36

Description

Step 1: implement the previous topology with the following peer relationships:

- Configure and IGP inside AS235
- R1 and R2 are eBGP peers. R3 and R4 are eBGP peers. R5 and R6 are eBGP peers.
- R2 and R3 are iBGP peers.
- R2 and R5 are iBGP peers.
- R2, R3, and R5 implement NEXT-HOP-SELF.
- Verify the IP routing table and BGP tables of each router.
- What is the routing knowledge of each router?

Step 2) Implement a route reflector server.

- Choose one strategic router to deploy the RR server.
- Repeat Analysis

Step 3) Implement a full mesh topology.

- Remove the configuration pertaining to the route reflector
- Connect a physical link between R3 and R5.
- R3 and R5 peer now using iBGP.
- Repeat Analysis

Step 4) Implement a logical full mesh topology.

- Shutdown or remove the physical connection between R3 and R5.
- Remove the configuration commands.
- Configure iBGP peering between R3 and R5 using the loopback addresses.
- Repeat analysis.

Exercise Route Reflection

- Implement the following topology.
- R3 must be a Juniper router and R5 must be a Cisco router. The others can be either.

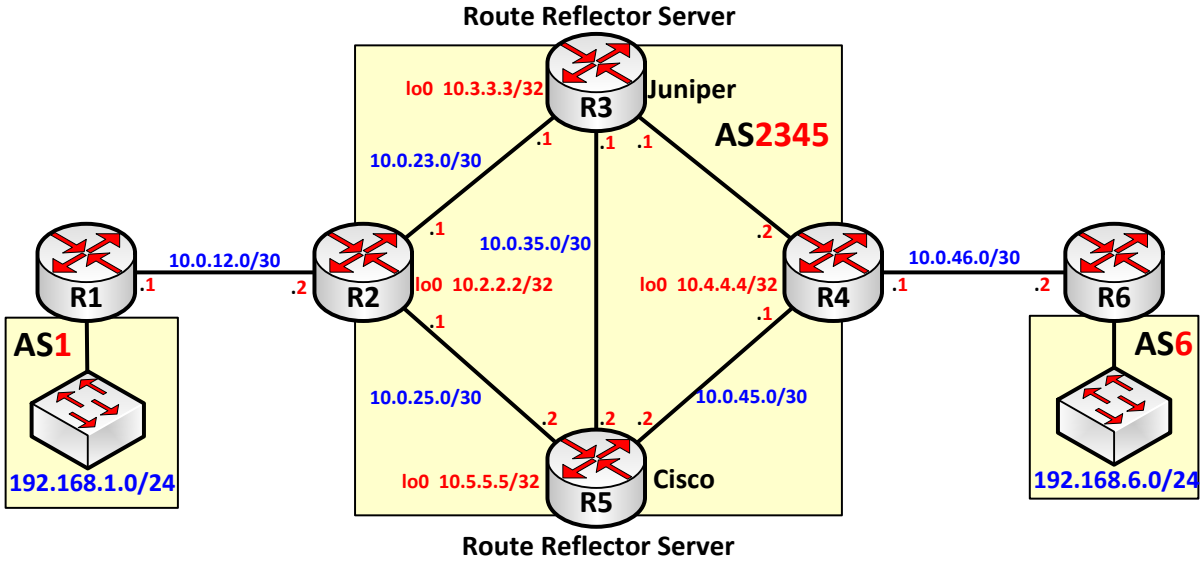


Figure 4.37

Description

- Implement an IGP inside AS2345.
- Implement the following BGP peering relationships:
- R1 and R2 are peers using eBGP. R1 advertises 192.168.1.0/24.
- R4 and R6 are peers using eBGP. R6 advertises 192.168.6.0/24.
- R2 and R3 are peers using iBGP.
- R3 and R5 are peers using iBGP.
- R5 and R4 are peers using iBGP.
- R2 and R4 are not peers.
- R2 and R5 are not peers.
- R3 and R4 are not peers.
- R3 is a route reflector. Routers R5 is its only RR client.
- R5 is a route reflector. Routers R4 is its only RR clients.

Analysis

- Collect all the routing tables and BGP tables in AS2345.
- Create a descriptive map of the routes with their valid paths and next hop paths.
- Look especially into the data from routers R3 and R5.
- Investigate if the external routes are reachable with the most efficient path.
- Fix the network by implementing iBGP where it is needed.

Exercise iBGP and Route Reflection

- Implement the following network. Use iBGP and route reflection where needed.
- You must consider the need for redundancy here. What happens when one of the internal links fails?

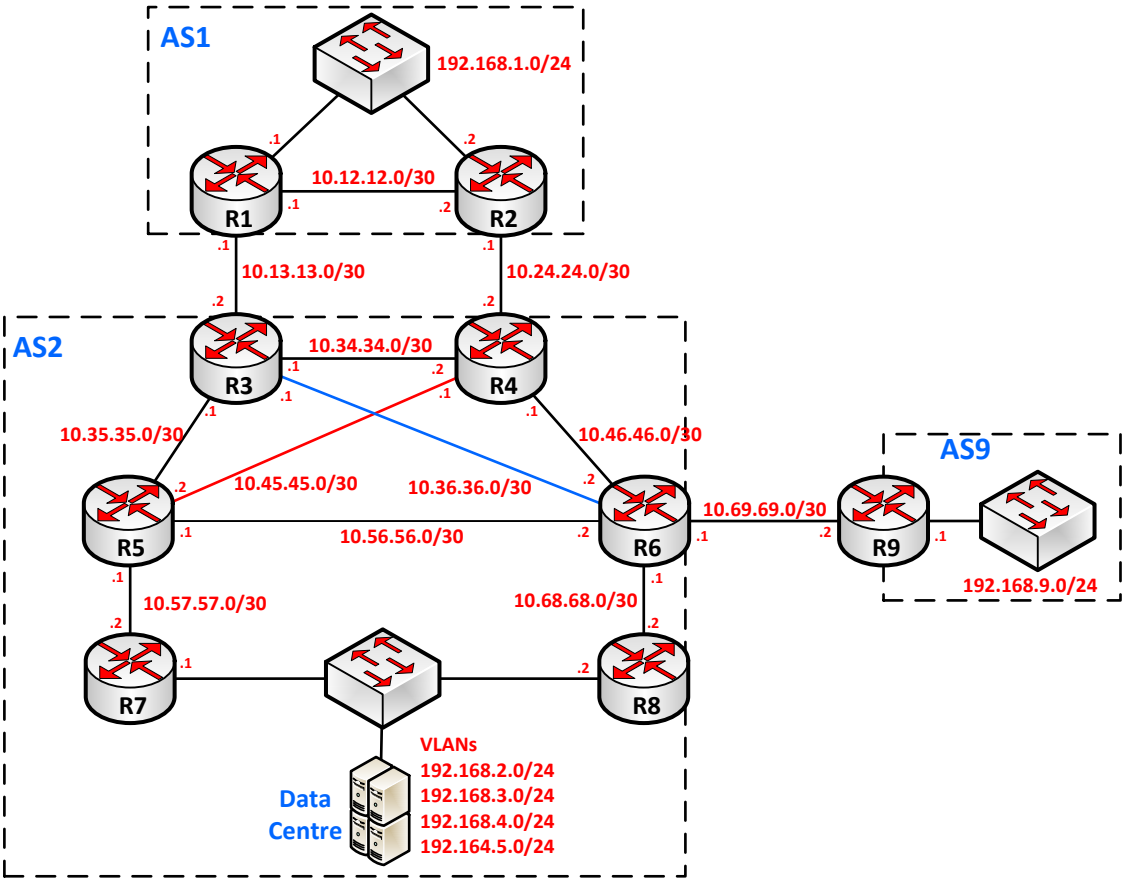


Figure 4.38

Exercise iBGP and Route Reflection

- Implement the following network. Use iBGP and route reflection inside AS5.

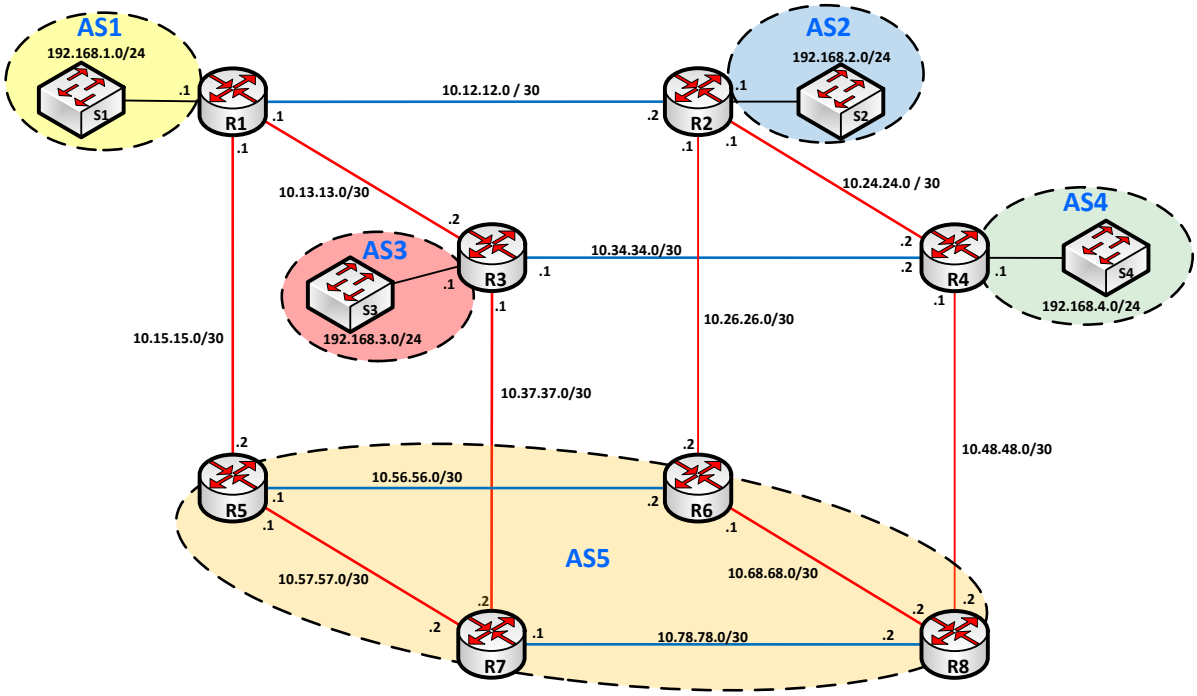


Figure 4.39

Exercise iBGP and Route Reflection

- Implement the following network. Use iBGP and route reflection wherever needed.

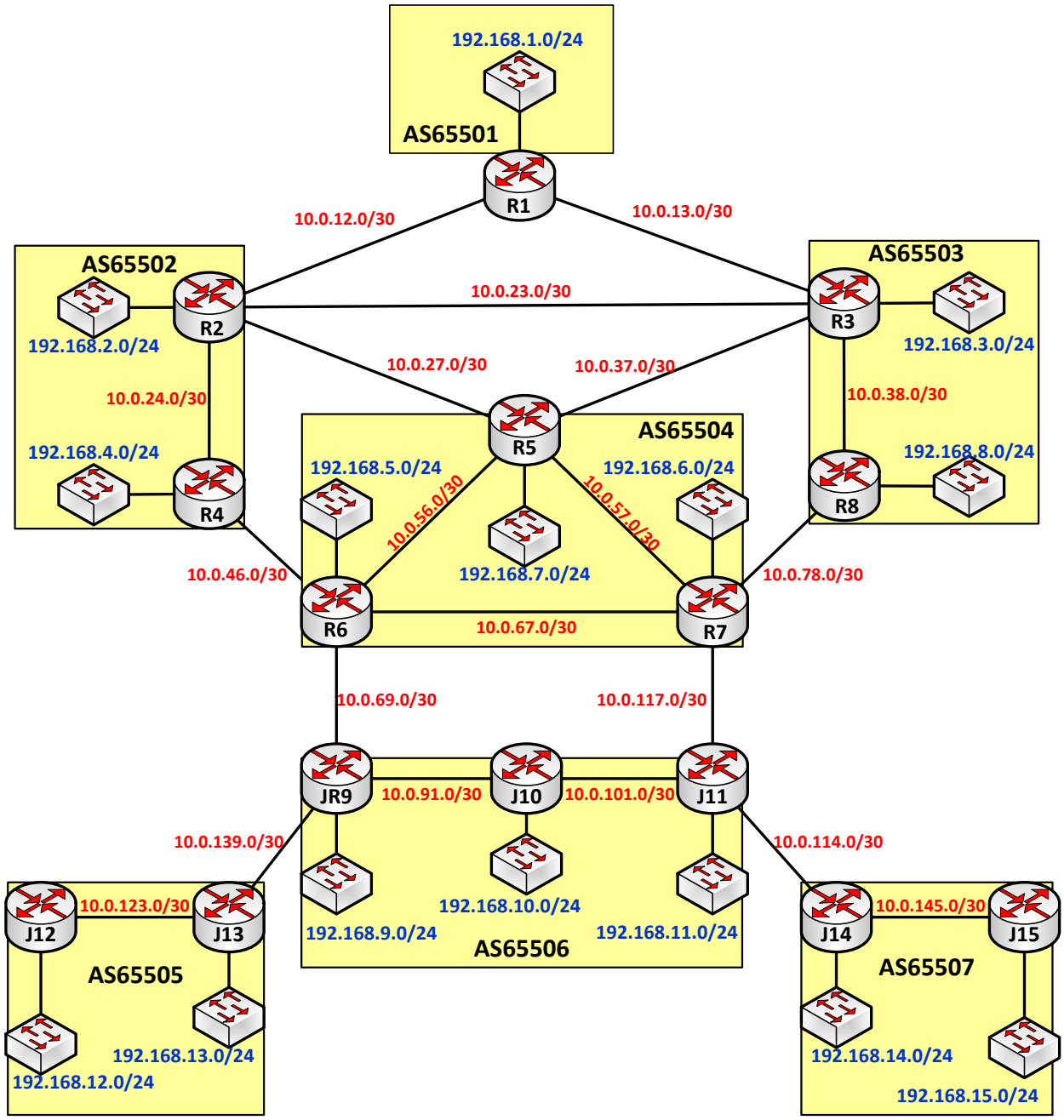


Figure 4.40

Chapter Reference

[1] R. Chandra. E. Chen. 2000. RFC-2796. BGP Route Reflection. An Alternative to Full Mesh IBGP. https://tools.ietf.org/html/rfc2796

CHAPTER 5

Techniques for BGP Policies

When having to select a path to a destination, BGP follows several well established steps of an algorithm. Nevertheless, the main point of using BGP is route manipulation by policies. In this regard, BGP is a unique routing protocol because it enables network administrators to write and enforce routing policies.

This chapter begins with an exploration of BGP's decision making default process. Then, the basic BGP routing engine is explored from the view point of the implementation of polices at a basic level. In the process, the fundamental tools and techniques for Cisco IOS and Juniper JUNOS are introduced.

BGP Default Decision Process

A BGP speaking router might have several entries for the same route in the BGP database but only the best entry will be presented for promotion into the IP routing table. The default behaviour of BGP is to follow a checklist to make the decision of choosing the best path to a route destination. The following network topology has been designed specifically to test the behaviour of the BGP algorithm. There is only basic peering without policy implementation.

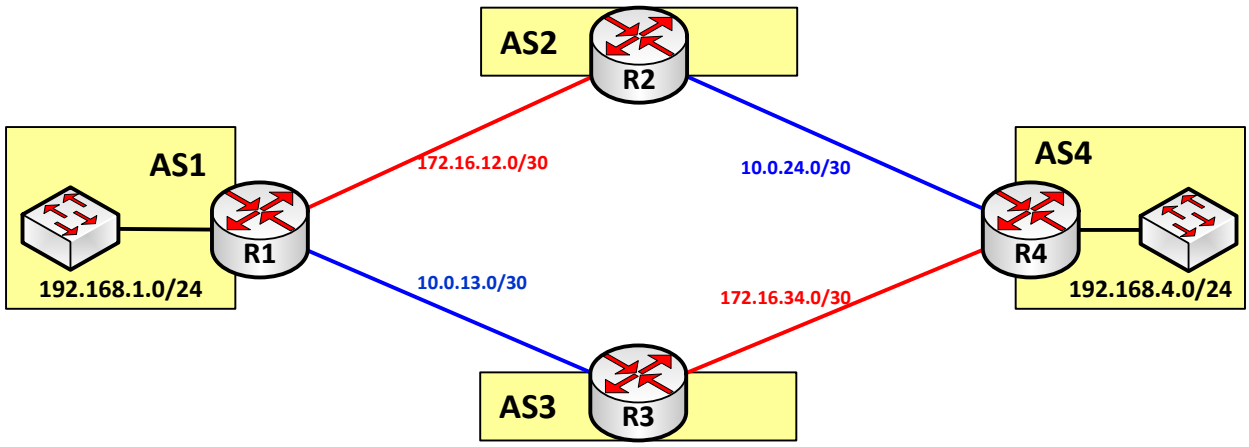


Figure 5.1

Once the configuration is completed, the routers BGP processes are re-started and the reachability is tested by trace-routing from R1 to R4. It is important to clarify now, that configuring the routers in different order can yield to a different logical routing topology. The following diagram contains the information needed to perform the analysis. Routers R1 and R4 are the only ones that advertise prefixes to focus the analysis only on them. The BGP databases entries are shown and the best routes are indicated with the symbols *> since the configuration was done in Cisco IOS.

A traceroute toward 192.168.4.1 is sent from R1. The traceroute is sourced from 192.168.1.1 so that R4 can reply back. Traceroute, by default, uses the outgoing interface of the device that is sending and that would be 172.16.12.1 in this case. The point to point DMZ links are not advertised by anybody, nor should they be.

R1# show ip bgp

Network	Next Hop	Metric	Loc-Pref	Weight	Path
*> 192.168.1.0	0.0.0.0	0		32768	i
*> 192.168.4.0	172.16.12.2			0	2 4 i
* 192.168.4.0	10.0.13.2			0	3 4 i

R4# show ip bgp

Network	Next Hop	Metric	Loc-Pref	Weight	Path
*> 192.168.1.0	172.16.34.1			0	3 1 i
* 192.168.1.0	10.0.24.1			0	2 1 i
*> 192.168.4.0	0.0.0.0	0		32768	i

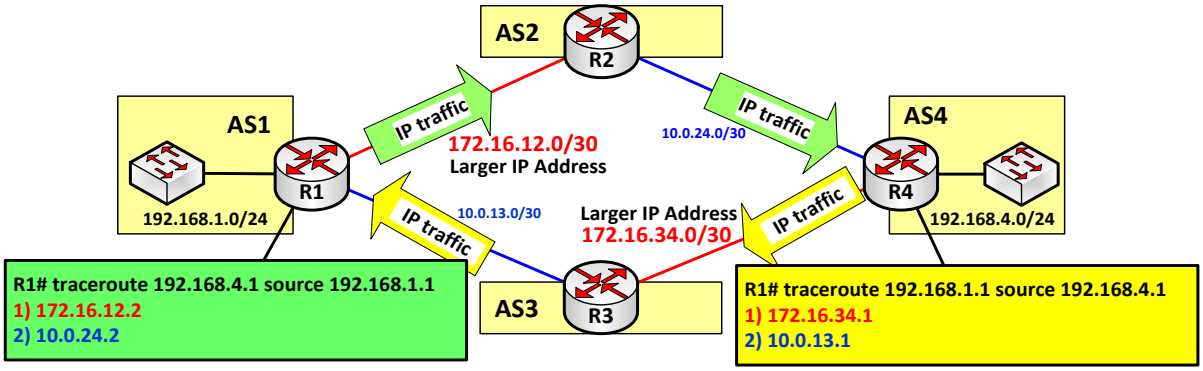


Figure 5.2

Router R1's BGP routing table shows that prefix 192.168.4.0 is reachable via the next hop of 172.16.12.2. So the traceroute message is sent to router R2 and from there it arrives to R4 on the 10.0.24.2 interface. Router R4 receives the request and prepares a reply by looking up the best path to the destination which for the reply is 192.168.1.1. In its BGP table, *> 192.168.1.0 appears with a next hop of 172.16.34.1 or going to R3. The result is that the paths forward and back are different. This is really not a grave problem because IP was precisely designed to deliver data in a connectionless way and that is what is happening here. The data is being transmitted and received anyways. However, for troubleshooting purposes, this could lead to

confusion for the person doing the traceroute. After all, the idea is to test the same path both ways. Also, if some traffic conditions were being implemented, like services guarantees or commercial agreements; that would be hard to do.

This network topology demonstrates that BGP has an inner algorithm that calculates the best paths and from the results obtained, it is evident that this algorithm is local to the position of the router doing the calculation. A different path forward and back would not likely result with an IGP routing protocol like OSPF or RIP. Then the question is why did BGP calculate the best paths as it appears in the previous network?

The BGP algorithm follows by default the steps of a check-list to make a decision. The diagrams below present each decision step together with an example case to illustrate the explanation.

Step 1) A BGP will always consider if the NEXT_HOP is reachable or not before anything else. In the example case, the prefix is advertised by router R2 to router R1. This router knows where the NEXT_HOP is located, so R1 can install and use the route prefix. However, when R1 advertises the prefix to R1b using iBGP, it must overwrite the NEXT_HOP to itself. Otherwise, R1b does not know where the NEXT_HOP is located and it would drop the route advertisement.

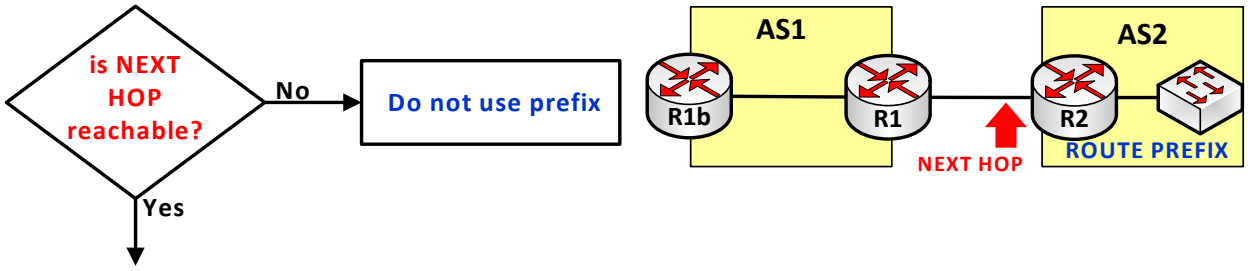


Figure 5.3

Step 2) If the NEXT_HOP is reachable from step 1, then BGP considers the WEIGHT if it is a Cisco router. WEIGHT **is not** a BGP attribute but a factor proprietary to Cisco. The weight is set up by the network operator and it is used to implement preference for a path that has the heavier weight.

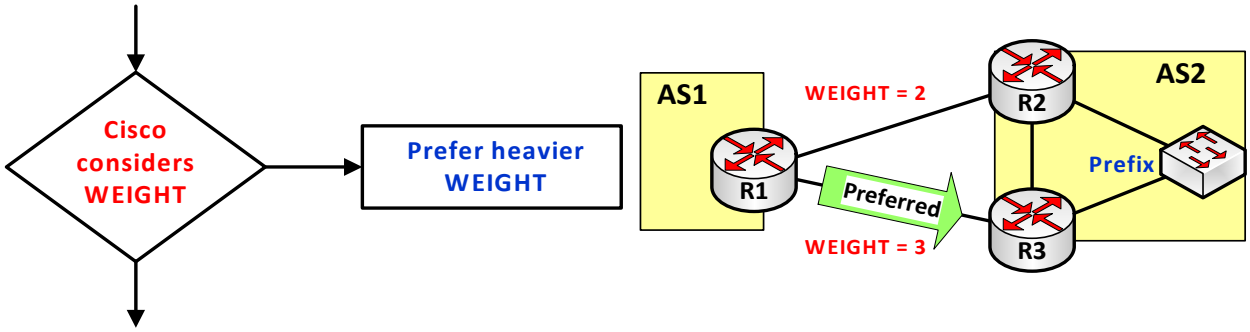


Figure 5.4

Step 3) This would be step 2 for a non-Cisco router. In the example case, router R1 is inside the routing domain and it speaks iBGP with the AS border routers (ASBRs). The routing policy in this domain implements the use of the attribute LOCAL PREFERENCE where the highest number is the best. The path with LOCAL_PREF of 300 has a higher preference than 200. As a result, R1 will send the traffic to the destination via the preferred path with LOCAL_PREF of 300.

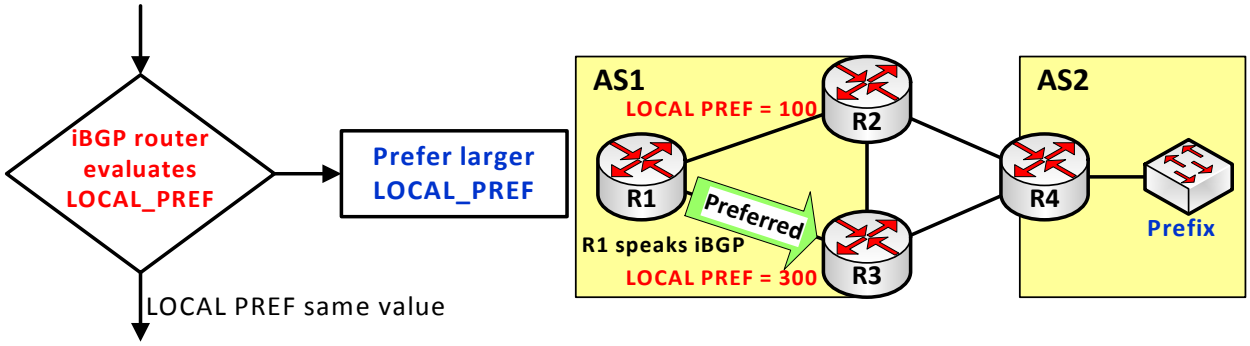


Figure 5.5

Step 4) Now, the BGP process evaluates the length of the AS_PATH attribute. The shortest path to a destination, measured in counting AS, is considered to be the best. Router R1 receives advertisements for the same prefix from two neighbours, so the prefix appears twice in the BGP table. When the AS_PATH length is evaluated there are two choices; AS2 and then AS4 or AS3 then AS4. Both have the same length so this is a tie. In this particular topology, the AS_PATH evaluation does not determine what path to use and the evaluation process must continue to the next step.

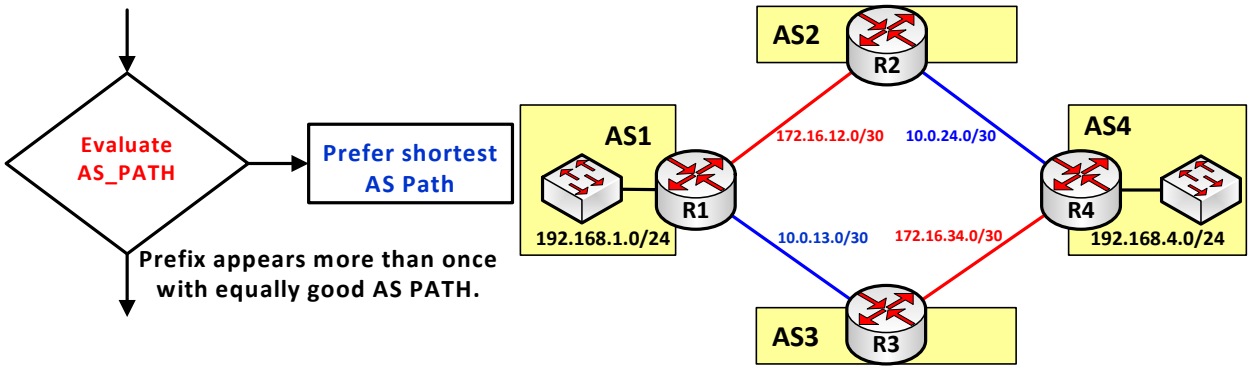


Figure 5.6

Step 5) The ORIGIN attribute attached to the prefix advertisement is used to choose a path to the destination now. An ORIGIN of IGP is preferred above all, and then there is no other choice than ORIGIN of incomplete. In the shown topology, both paths offer the same ORIGIN characteristics, so there is no decision yet.

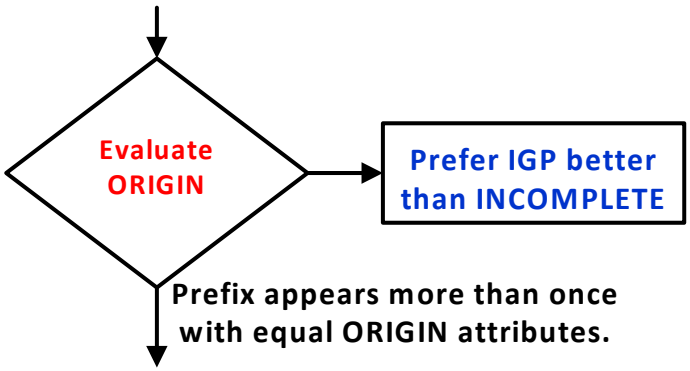


Figure 5.7

Step 6) The next attribute to be considered is the Multi-Exit-Discriminator or MED. The value of MED acts like a metric so the lowest, the better. In the example diagram, the AS2 border router

advertises the prefix to the router ASBR in AS1. The top ASBR advertises the prefix with a lower metric (50) than the bottom router (100). The result is that the AS1 router prefers the path that leads to the top ASBR.

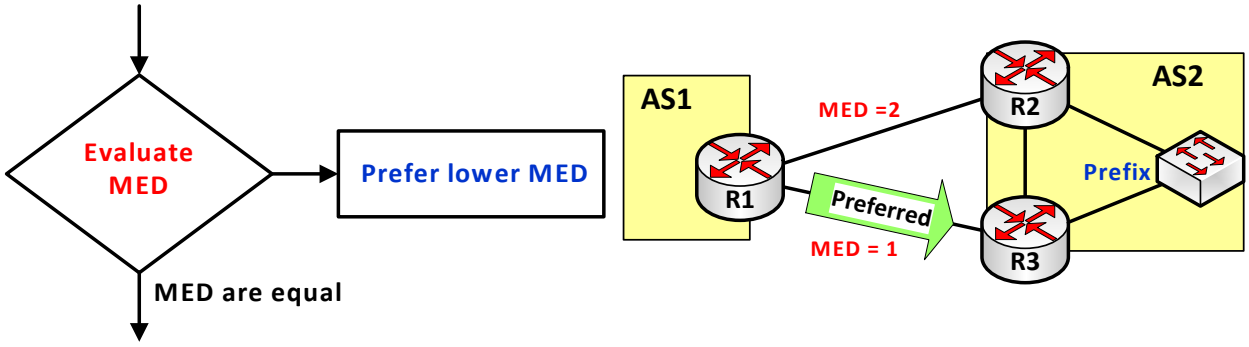


Figure 5.8

Step 7) If the MED is not a factor to make a decision or if the MED is the same via different path to the peers, then the internal metric is evaluated. For example, router R1 uses the shortest path metric to the exit point toward the prefix destination.

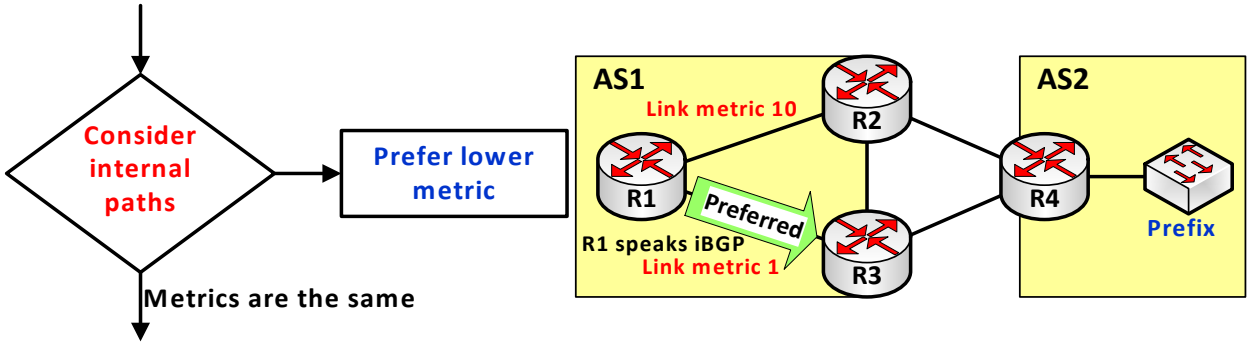


Figure 5.9

Step 8) If there are several equal paths to the same network destination, then the link with the higher IP address would be preferred.

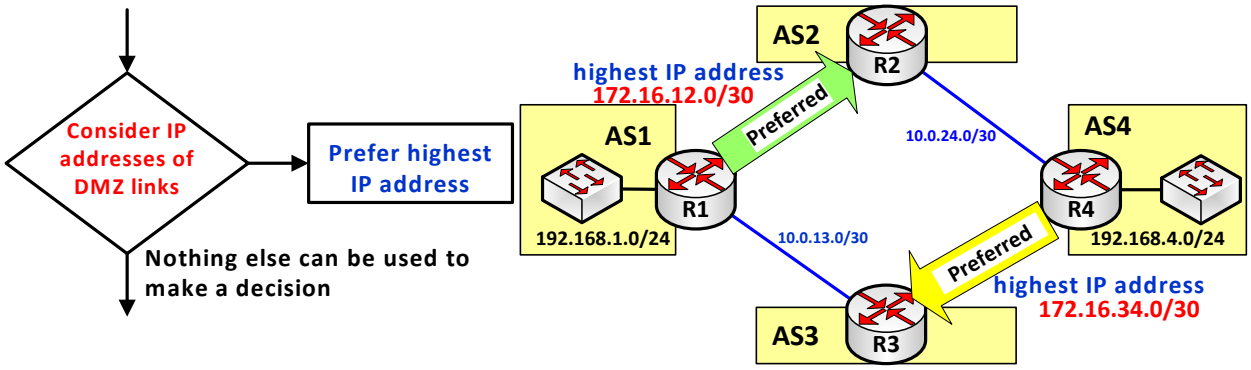


Figure 5.10

Cisco has a proviso that the first equal path that gets installed, it will remain until the routers are clear again. In our topology, R1 chose the path via R2 to reach 192.168.4.0/24 because of this rule. R4 did the same from its viewpoint to reach 192.168.1.0/24.

Step 9) If all the previous choices have been considered and no decision has been reached, then the BGP routers will have to evaluate the router-id of the advertising routers and prefer the route that is advertised by the lowest router ID neighbour.

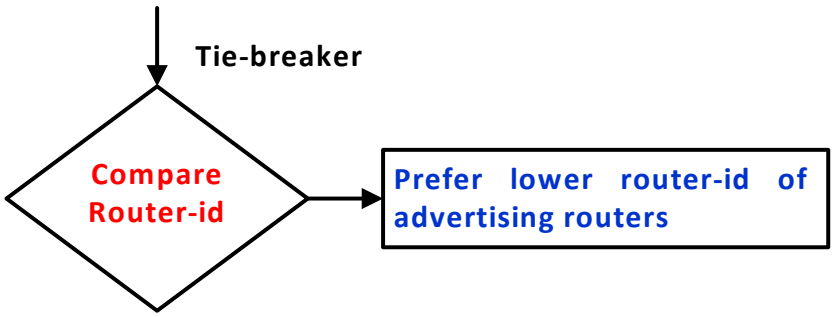


Figure 5.11

Thus, it is important that this is properly included into the configuration tasks. In JUNOS, it is a requirement that the router-id be manually assigned by the administrator typically by matching the IP address of the loopback interface. In Cisco IOS, this is optional and that leaves open a number of combinations. If the router-id is not manually assigned, the router will pick up an identification depending of the configuration sequence. For example:

- If a loopback interface has been configured already, that would be used by the router.
- Otherwise, the router will use the IP address of the outgoing interface to the neighbour.
- If the router is re-booted, then the loopback address will be considered first.

In any case, once that a router-id is selected, that will be used to identify the router doing the advertisement.

These are the steps of the BGP engine by default but this decision process can be modified with the introduction of BGP Policies. In fact, most of the workload of implementing BGP is setting up policies.

Modifying BGP Default Behaviour with Policies

The routing information that BGP propagates throughout the Internet and the resulting IP traffic flows are the products of the policies implemented at various autonomous systems. Depending of the traffic orientation, the policies are applied either **inbound** to a router or **outbound** from a router.

The inbound policy is designed to accept the desired prefixes and to add or modify the attributes that accompany the routing information. It is the decision of the network administrators to specify what routing information is going to be accepted by the autonomous system. The general mechanics of the **inbound policy** is illustrated in the next figure.

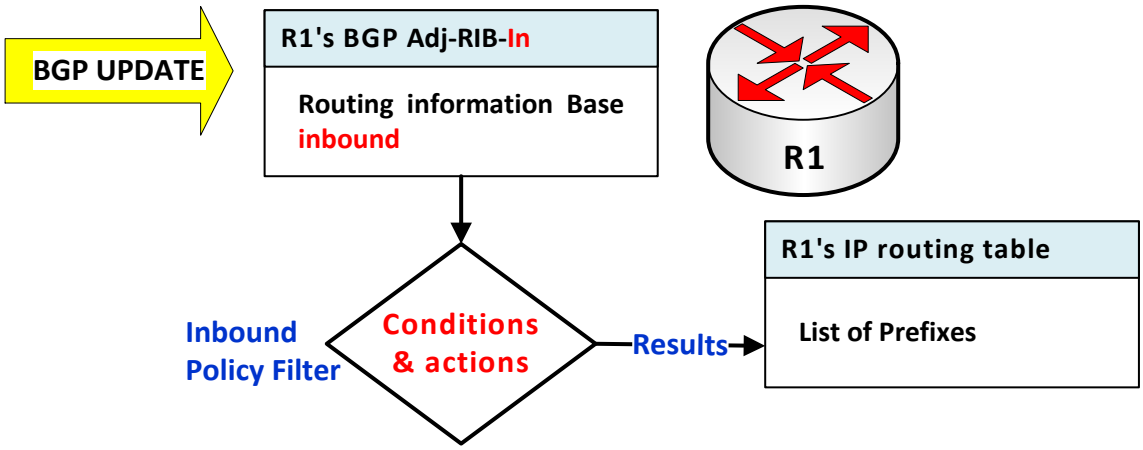


Figure 5.11

When a router receives a BGP UPDATE message, it processes the advertised IP route prefixes against the conditions specified in the inbound applied policy. If matches to the conditions are found then actions are taken like:

- Rejecting some prefixes.
- Accepting the prefixes as they are.
- Accepting the prefixes while changing some attributes.

The prefixes that pass thru the policy are stored in a BGP database Adjacent-RIB-in. This is typically a software instance [2] and it is linked to the adjacency process that relates to the BGP neighbour. Depending of the situation, there could be one policy per neighbour or the same policy could be applied to a group of neighbours that share the same traffic interest.

The same prefix can appear several times in the database, but only the best choice (*>) is presented to the Internet Protocol IP as a candidate to be in the IP routing table. IP considers if there is a better source advertising the prefix although (external) BGP has a better preference than most IGP. If the prefix is accepted by IP, then it appears in the routing table and data can be sent to that destination. Also, the prefix installed in the IP routing table can be considered to be re-advertised to another neighbour but that would be part of the outbound process which is explained next.

Outbound BGP Policies are used when a router advertises IP prefixes to another router. These policies control the sending of a subset of the routes present in the routing table or the sending of the routing information with certain attributes so they can be used to match and then execute routing actions in the receiving end. There should be an outbound BGP policy specific per each kind of peering relationship. The general structure of the BGP outbound policy engine looks like this:

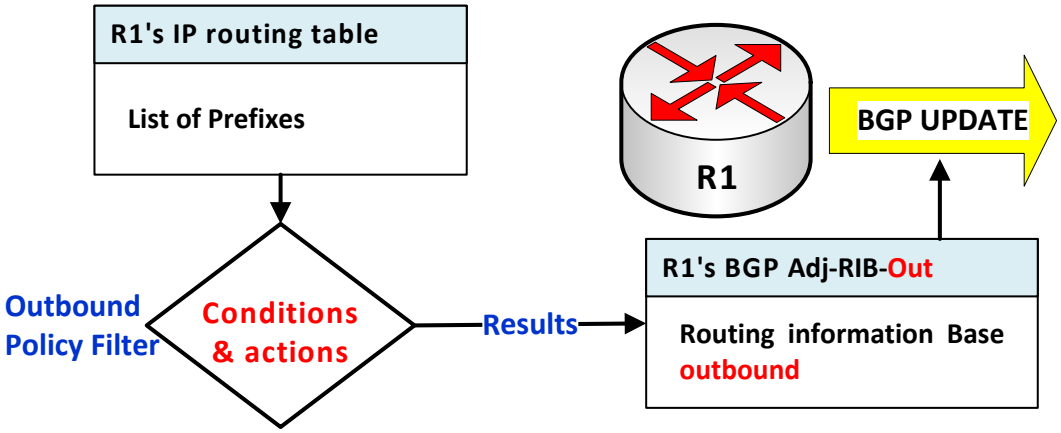


Figure 5.12

In this previous figure, the router reads the IP prefixes from the routing table, and then they are evaluated against the conditions written in the outbound BGP policy. The outbound policy main objective is to take actions like:

- Blocking some prefixes from being advertised.
- Allowing some prefixes to be advertised without changing any property.
- Advertising some prefixes, but changing some attributes.

First of all, the prefixes have to be present in the IP routing table. The routing information is collected and compared to the conditions written in the policy. If there are matches, then some actions are executed. The resulting subset of routing information is stored in an Adjacent Routing Information Base Adj-RIB-out. This is also a software instance like the Adj-RIB in. From this store, the routing information is collected and formatted into BGP UPDATE messages to be sent to a neighbour router.

The complete processing of the BGP routing engine is summarized in the next diagram. R2 has an inbound policy that rules its relationship with R1. When R1 sends an UPDATE, the routing information passes thru such policy and the resulting subset of routes is to be installed in the BGP database. The best valid routes are then presented to the IP protocol for inclusion into the routing table.

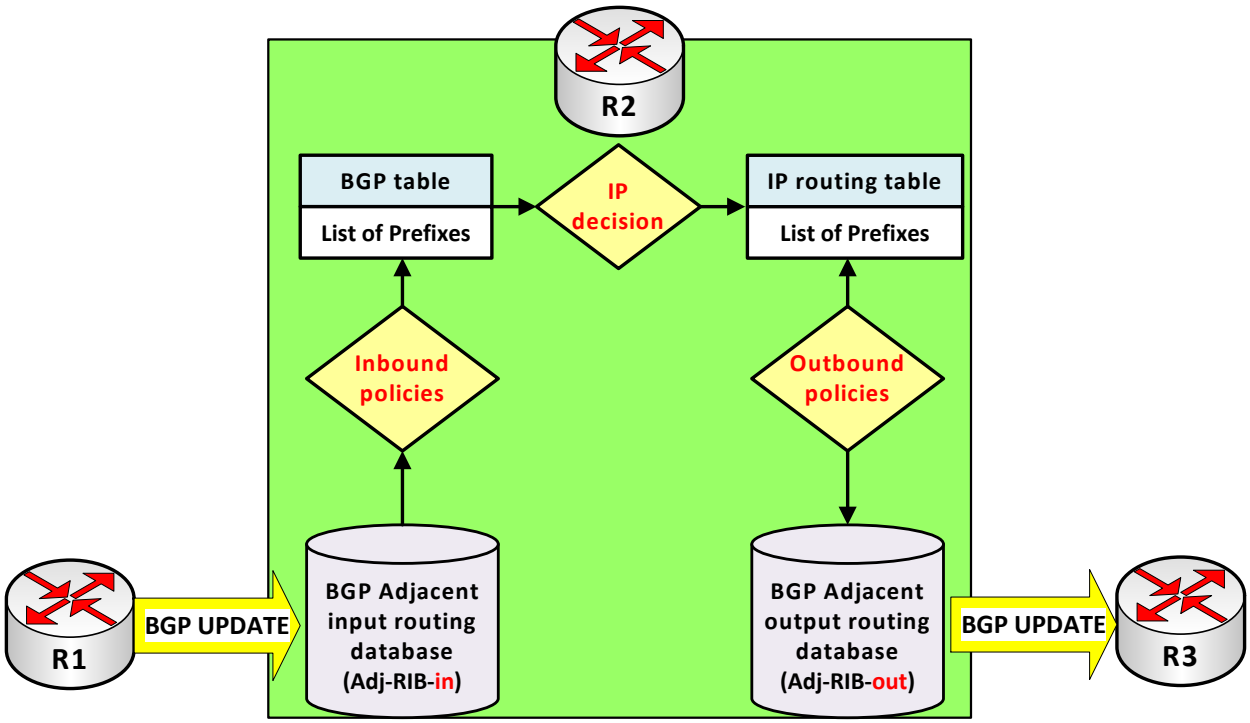


Figure 5.19

On the other side, R2 has an outbound policy that rules the routing content that is advertised to R3. The IP routes are obtained from the IP routing table, then passed thru the outbound policy and the result will be formatted and send in BGP UPDATES.

Implementing Policies in BGP: the basic tools

Both inbound and outbound policies have common factors such as:

- Accepting or rejecting prefixes
- Matching properties
- Modifying attributes

Implementing each one of these factors requires the use of tools or techniques, specifically:

Action	Cisco IOS	Juniper JUNOS
Accepting or rejecting prefixes.	Access-list	Route-filter
	Prefix-list	Prefix-list
	Distribute-list	
Matching properties or attributes.	Regular Expressions	Regular Expressions
	Route-map	Policy-statement
Modifying properties or attributes.	Route-map	Policy-statement

Table 5.1

Regular Expressions

Regular expressions are character matching statements that have been long used in programming languages and UNIX/Linux like operating systems. The basic idea is to arrange a combination of special symbols to match a string of characters. In BGP, the most typical use of regular expressions is to find or to match routes advertised by an autonomous system. For example, a network engineer would like to know the prefixes advertised by Sheridan College in Ontario, Canada. The full Internet Routing Table (on July 2017) shows the following:

```
felix@S144b> show route terse
645,050 IPv4 routes
```

Obviously, looking through over 600 thousands routes would be unproductive. Instead, the network engineers can look-up the AS number for Sheridan College and use a regular expression to match it. The result can be seen in this example done in a Juniper router:

```
felix@S144b> show route aspath-regex .*5664$ terse
A Destination          P Prf    Next hop       AS path
* 142.55.0.0/16        B 170    >10.254.12.1   65144 123456 812 174 5664 i
* 142.240.254.0/24     B 170    >10.254.12.1   65144 123456 812 174 5664 i
```

Table 5.2

Which returns only two prefixes sourced by AS 5664 and that is exactly what the network engineer wanted to know. Even more, if the regular expression can be used with the "show command" then the same expression can be used inside a policy to match the source of the routes and then apply some conditions. The following network will be used to study the use of regular expressions.

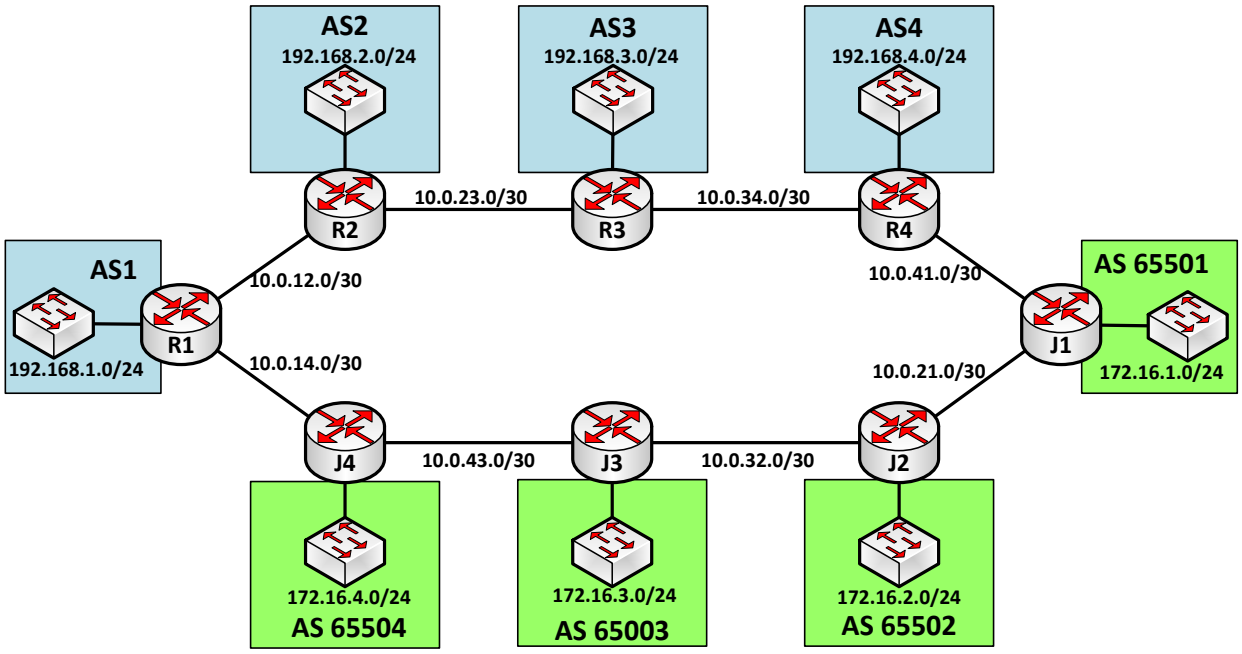


Figure 5.13

The topology above yields this routing table in router R1. The network has been purposely designed to have all the prefixes 172.16.x.0 in autonomous systems in the range 65,5xx deployed in Juniper routers. The prefixes that have 192.168.x.0 belong to AS numbers 1, 2, 3, 4 deployed in Cisco IOS routers.

Condition	Network	Next Hop	BGP AS-PATH					
* valid	172.16.1.0/24	10.0.12.2		2	3	4	65501	i
*> best	172.16.1.0/24	10.0.14.2		65504	65503	65502	65501	i
*> best	172.16.2.0/24	10.0.14.2			65504	65503	65502	i
*> best	172.16.3.0/24	10.0.14.2				65503	65502	i
*> best	172.16.4.0/24	10.0.14.2					65504	i
*> best	192.168.1.0/24	0.0.0.0						i
*> best	192.168.2.0/24	10.0.12.2					2	i
*> best	192.168.3.0/24	10.0.12.2				2	3	i
* valid	192.168.4.0/24	10.0.14.2	65504	65503	65502	65501	4	i
*> best	192.168.4.0/24	10.0.12.2			2	3	4	i

Table 5.3

Let's explore some regular expressions with the help of the Cisco IOS command "show ip route regexp" since R1 is Cisco. But before, keep in mind the following rules:

- In regular expressions, the beginning of the string of characters is the left-most character.
- On the other hand, the beginning of the AS-PATH attribute (the source of the route) is read from the right-most number.

Regular Expressions Examples

Statement: show all the prefixes that have an AS-PATH of ASN 2 only.

```
R1#show ip bgp regexp ^2$
Network          Next Hop      Path
*> 192.168.2.0   10.0.12.2     2 i
(Notice that AS2 is a neighbour of AS1)
```

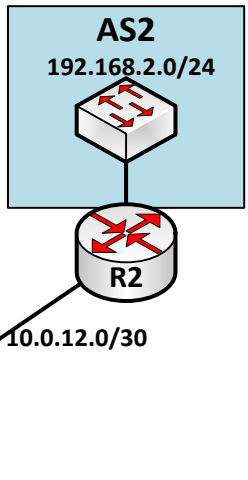


Figure 5.14

Explanation:

- The string of characters is read from left to right.
- The symbol **$** means end of the string, but that is the beginning of the AS-PATH.
- The symbol **^** means the beginning of the string which is the end of the AS-PATH.
- This is an AS-PATH that begins and ends in number 2.
- In conclusion, the prefix 192.168.2.0/24 is the only prefix advertised by AS2 that reaches AS1 with no other AS, but AS2 in the path.
- Notice: this command returned an output because AS2 is directly connected to AS1, see the next case to compare.

Statement: show all the prefixes that have an AS-PATH of ASN 3 only.

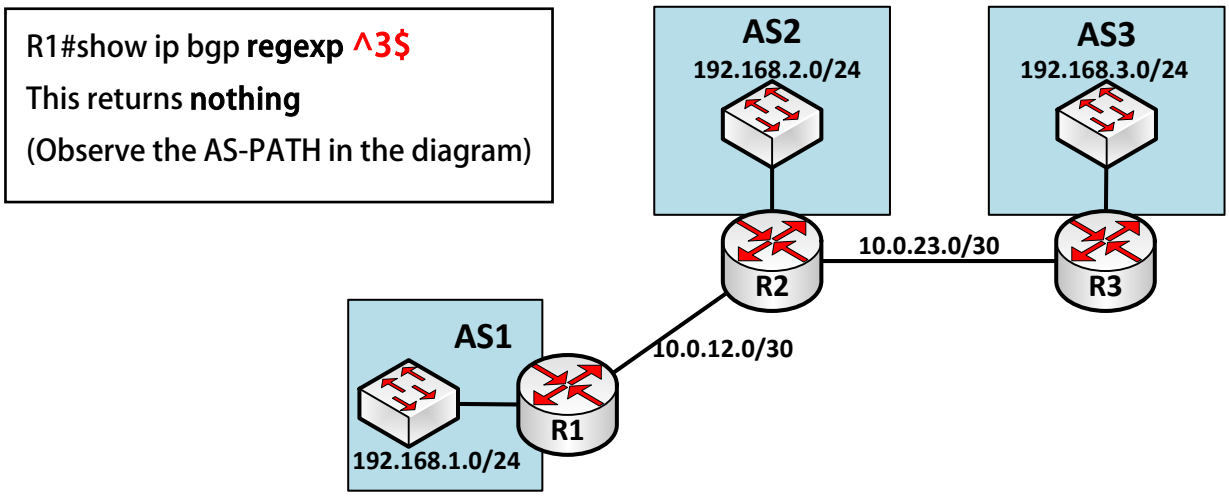


Figure 5.15

Explanation:

- The command returns nothing even though there is a route (192.168.3.0/24) sourced from AS3.
- The reason is that the regular expression ^3$ matches the routing information that has only AS3 in the AS path and since the command is being issued in AS1-R1 that is not a possible match.
- The $ sign is the beginning of the string and the caret ^ sign is the end of the string.
- Basically, it is either the wrong question or in the wrong place.

Statement: show the prefixes that have a 3 at the beginning of the AS-PATH attribute. (Note: the AS-PATH begins from the right-most character)

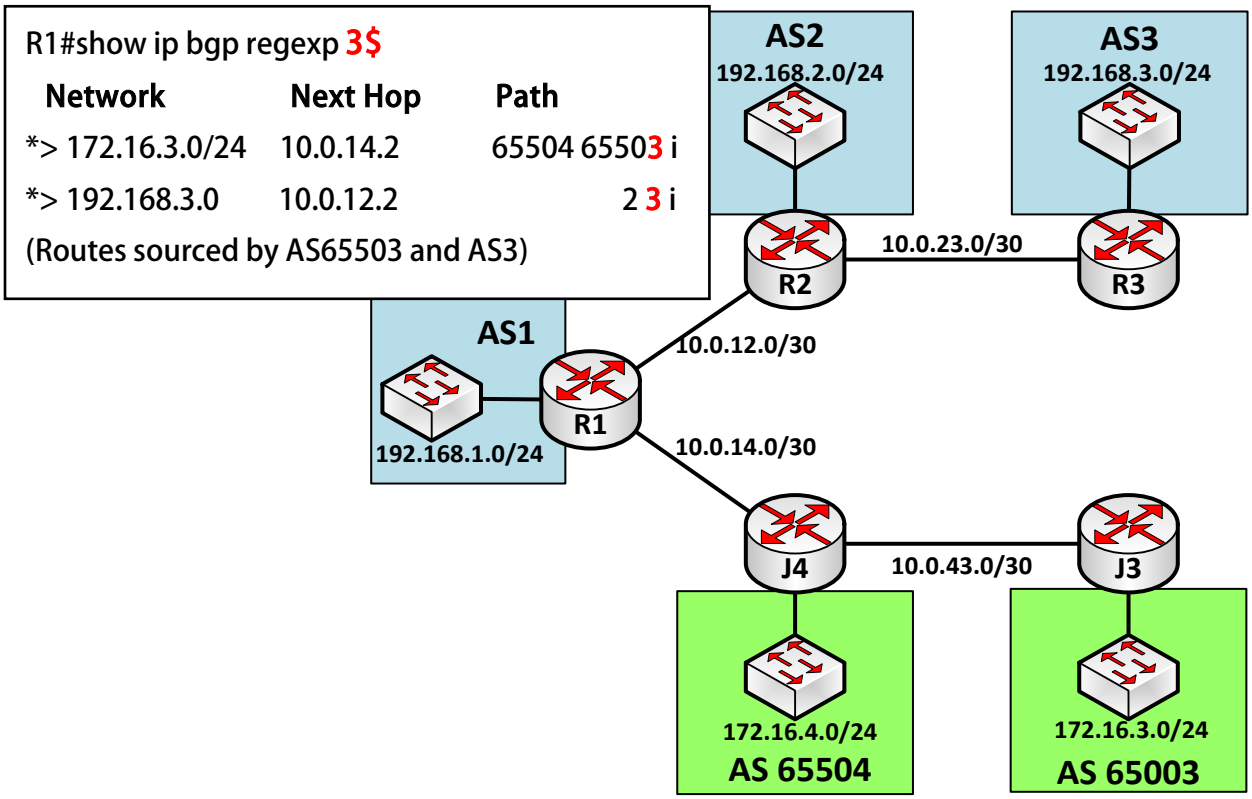


Figure 5.16

Explanation:

- The right-most character of the AS-PATH is matched by the expression 3$. The string's leftmost side is open so that matches all the strings that begin with 3 (reading from right to left) and that includes 65503. A more precise expression could be "show only the routes installed by AS3 with any path". Let's try that one:

Statement: show the prefixes that are sourced by ASN3 only (with any path)

```
R1#show ip bgp regexp _3$
  Network          Next Hop          Path
*> 192.168.3.0      10.12.12.2        2 3 i
(Sourced by AS3 with any path)
```

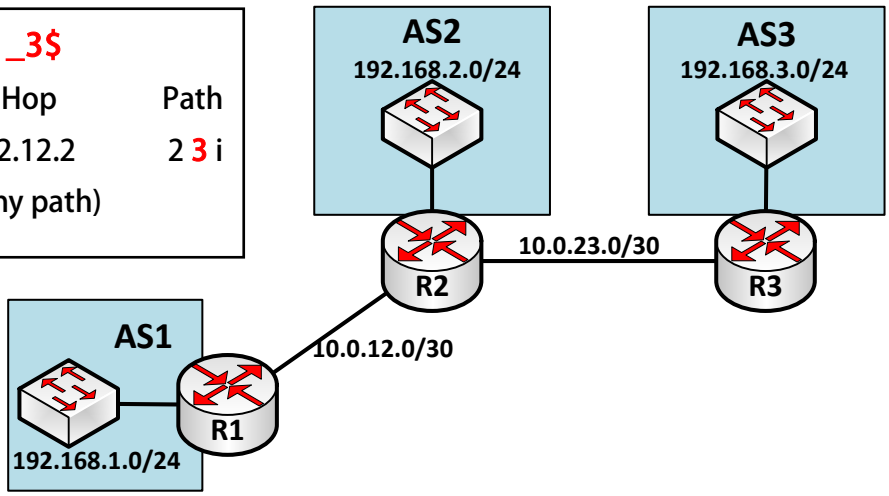


Figure 5.17

Explanation:

- The right-most character of the AS-PATH string is matched by the expression 3$. The string's left side has an underscore character that means match a space. That gets AS3 and anything else in the path.

Statement: In R1, show the internal network prefixes to be advertised.

```
R1#show ip bgp regexp ^1$
This returns nothing
```

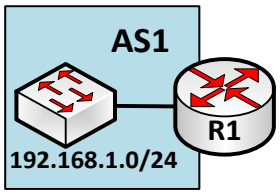


Figure 5.18

Explanation:

- This regular expression is correctly written but it makes no sense if it is issued in R1 since this router belongs to AS1 and there is the BGP rule that says "the AS number is stamped on the AS-PATH attribute when the UPDATE is sent to an external BGP router". In short, the AS_PATH for 192.168.1.0/24 is empty from the viewpoint of R1.
- In router R1, the correct expression "to match its own AS routes" would be:

```
R1#show ip bgp regexp ^$
  Network          Next Hop    Path
*> 192.168.1.0     0.0.0.0     i
```

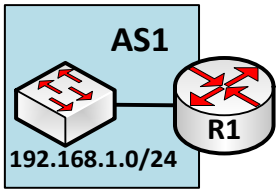


Figure 5.19

- This is a very helpful regular expression known as the "**empty path regular expression**" and it matches the prefixes that have no ASN in the AS-PATH hence they are internal routes in this AS domain. This expression is used in export policies and to control iBGP advertisements.

Statement: show anything that is injected to AS1 directly from AS2.

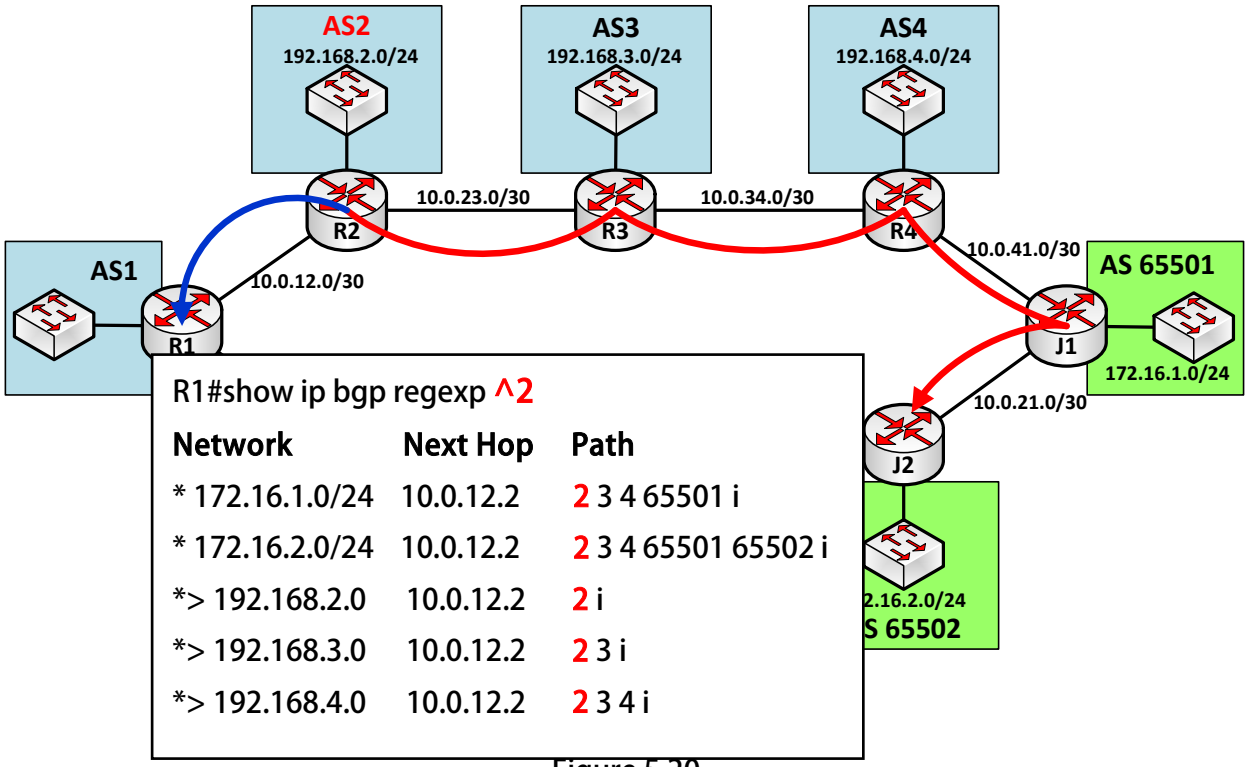


Figure 5.20

Explanation:

- The caret symbol to the left of the number 2 matches AS2 as the end of the AS-PATH attribute (or the beginning of the string).
- This command is issued in R1-AS1 and AS2 is a neighbour so AS2 is injecting these routes into AS1.

Statement: show the prefixes that have ASN 65503 somewhere in the AS-PATH.

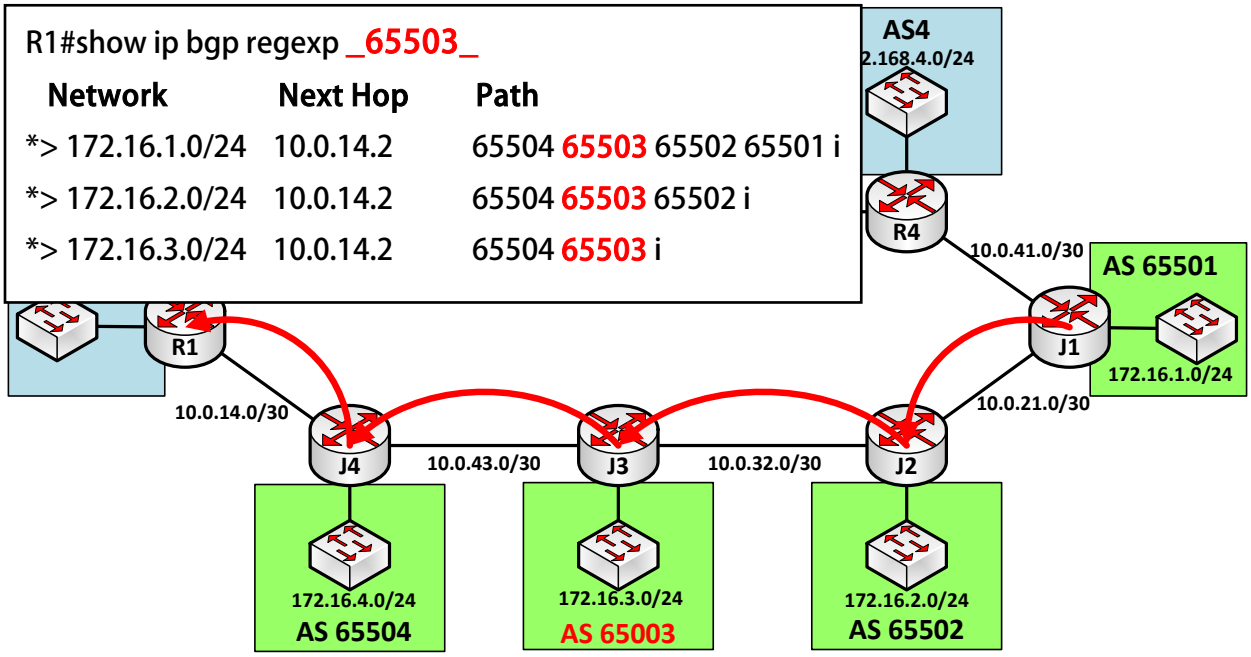


Figure 5.21

Explanation:

- The regular expression _65503_ could be read as any string of characters with a "blank space 65503 blank space" and all the combinations shown above comply with this condition.
- It does not matter where the ASN 65003 is located as long as it is present in the path.

Statement: show the prefixes that are sourced by an ASN that has 03 at the beginning of the AS-PATH sequence.

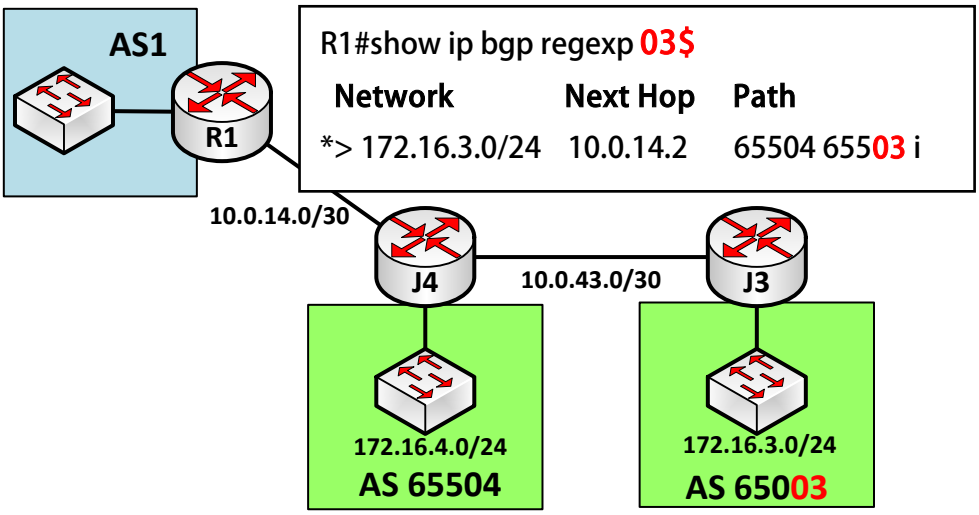


Figure 5.22

Explanation:

- The characters matched by the dollar sign are 03 and only the string 65503 complies with that condition.

Statement: show the prefixes that are sourced by ASNs that have 50any# at the beginning (right-most portion) of the AS-PATH sequence.

```
R1#show ip bgp regexp 50.$
Network          Next Hop     Path
* 172.16.1.0/24   10.0.12.2                    2 3 4 65501 i
*>172.16.1.0/24   10.0.14.2    65504 65503 65502 65501 i
*> 172.16.2.0/24  10.0.14.2          65504 65503 65502 i
*> 172.16.3.0/24  10.0.14.2                65504 65503 i
*> 172.16.4.0/24  10.0.14.2                      65504 i
```

Table 5.4

Explanation:

- The dot symbol means "any single number" that would be in the position of the dot, like in this case the numbers 1, 2, 3 and 4.
- All the AS numbers that have the pattern 50 and then any single number as the sourcing AS are caught here.

Statement: show the prefixes that have 50any# at the beginning of the AS-PATH string and then any other AS number.

```
R1#show ip bgp regexp 50.*$
Network          Next Hop      Path
* 172.16.1.0/24   10.0.12.2                    2 3 4 65501 i
*>172.16.1.0/24   10.0.14.2     65504 65503 65502 65501 i
*> 172.16.2.0/24  10.0.14.2           65504 65503 65502 i
*> 172.16.3.0/24  10.0.14.2                 65504 65503 i
*> 172.16.4.0/24  10.0.14.2                       65504 i
* 192.168.4.0     10.0.14.2    65504 65503 65502 65501 4 i
```

Table 5.5

Explanation:

- The regular expression 50. has been explained in the previous example.
- The addition of the symbol * takes other combinations.
- The whole expression takes the ASN with the characters 50 and any other combination present to the right of the string including the number 4 in the last line.

Statement: show the prefixes injected from my neighbour AS2 that were sourced by AS3.

```
R1#show ip bgp regexp ^2_3$
  Network          Next Hop    Path
*> 192.168.3.0      10.0.12.2   2 3 i
```

Table 5.6

Explanation:

- The dollar sign to the right of number 3 matches AS3 as the origin of the AS-PATH and the caret symbol to the left of the number 2 matches AS2 as the end of the AS-PATH string.
- The symbol underscore matches the blank space between the two AS numbers.

Statement: show anything that is injected to AS1 directly from AS2.

```
R1#show ip bgp regexp ^2_
Network          Next Hop     Path
* 172.16.1.0/24   10.0.12.2    2 3 4 65501 i
*> 192.168.2.0    10.0.12.2    2 i
*> 192.168.3.0    10.0.12.2    2 3 i
*> 192.168.4.0    10.0.12.2    2 3 4 i
```

Table 5.7

Explanation:

- The symbol underscore matches a blank space and the expression is open to the left, so it takes everything else.

Statement: show the routes transmitted from my neighbour AS2 with any AS source.

```
R1#show ip bgp regexp ^2 .*
Network          Next Hop     Path
* 172.16.1.0/24   10.0.12.2    2 3 4 65501 i
*> 192.168.2.0    10.0.12.2    2 i
*> 192.168.3.0    10.0.12.2    2 3 i
*> 192.168.4.0    10.0.12.2    2 3 4 i
```

Table 5.8

Explanation:

- The left-most position of the string has to be 2 and that can only means that AS2 is a neighbour of AS1. Basically the string ends in 2.
- The symbol dot (.) means any single character and the symbol start (*) means zero or more sequences of the preceding symbol. Altogether, the symbols (.*) mean any combination of AS numbers from the beginning of the AS_PATH all the way to AS2.

Statement: show anything that is injected to AS1 directly from AS2, but that is sourced from any direct neighbour of AS2.

```
R1#show ip bgp regexp ^2_[0-9]*$
 Network          Next Hop         Path
*> 192.168.2.0    10.0.12.2        2 i
*> 192.168.3.0    10.0.12.2        2 3 i
```

Table 5.9

Explanation:

- The string begins with AS2 as the immediate neighbour of R1-AS1.
- There is an underscore symbol that matches the blank space in the AS-PATH.
- Then the [0-9]* means any combination of these numbers including 0. (try 1-9)
- The string closes with a $ sign so there are no more AS in the AS-PATH.

Statement: show anything that has the number 4 anywhere in the AS-PATH.

```
R1#show ip bgp regexp 4+
Network          Next Hop    Path
*> 192.168.4.0     10.0.12.2    2 3 4 i
* 172.16.1.0/24    10.0.12.2    2 3 4 65501 i
*> 172.16.4.0/24   10.0.14.2    65504 i
*> 172.16.3.0/24   10.0.14.2    65504 65503 i
*> 172.16.2.0/24   10.0.14.2    65504 65503 65502 i
*> 172.16.1.0/24   10.0.14.2    65504 65503 65502 65501 i
* 192.168.4.0      10.0.14.2    65504 65503 65502 65501 4 i
```

Table 5.10

Explanation:

- The symbol + after the 4 means to match any combination of the number 4 present in the string.
- This includes repetitions of the character in the string like in the route 192.168.4.0.
- This is a helpful expression to remember since the network operator just need remember some of the numbers in an ASN but not all.

Statement: show the routes that have either the numbers 2 OR 3 anywhere in the AS-PATH

```
R1#show ip bgp regexp 2+|3+
Network          Next Hop      Path
*> 172.16.1.0/24 10.14.14.2    65504 65503 65502 65501 i
*> 172.16.2.0/24 10.14.14.2    65504 65503 65502 i
*> 172.16.3.0/24 10.14.14.2    65504 65503 i
*> 192.168.2.0   10.12.12.2    2 i
*> 192.168.3.0   10.12.12.2    2 3 i
*> 192.168.4.0   10.12.12.2    2 3 4 i
```

Table 5.11

Explanation:

- The symbol + matches any combination or sequence of the preceding numbers.
- The symbol pipe (|) acts as an OR operator.
- The result is the AS-PATHs that have 2 or 3 in the AS-PATH including 65502 and 65503.
- What would it be the regular expression that gives only AS2 or AS3?

Statement: show the routes sourced by either AS2 or AS65502.

```
R1#show ip bgp regexp _2$|_65502$
 Network          Next Hop       Path
*> 172.16.2.0/24   10.14.14.2     65504 65503 65502 i
*> 192.168.2.0     10.12.12.2     2 i
```

Table 5.12

Explanation:

- This is a more specific expression that the previous one.
- The addition of the underscore symbols constrains the sequence to only include AS2 and 65502.

Statement: show the routes that have either AS2 anywhere in the path or AS65502 as the source.

```
R1#show ip bgp regexp _2_|_65502$
Network          Next Hop       Path
* 172.16.1.0/24   10.12.12.2     2 3 4 65501 i
*> 172.16.2.0/24  10.14.14.2     65504 65503 65502 i
*> 192.168.2.0    10.12.12.2     2 i
*> 192.168.3.0    10.12.12.2     2 3 i
*> 192.168.4.0    10.12.12.2     2 3 4 i
```

Table 5.13

Explanation:

- This expression catches the AS2 anywhere in any AS-PATH with the use of the underscore symbols.
- The use of the dollar sign affixes 65502 as a sourcing ASN.

Regular Expression in JUNOS

JUNOS also implements regular expressions to search both the IP routing table and the BGP routing table. There are some differences with respect Cisco IOS though.

Statement: in router J1, show the routes injected by the neighbour AS65502

```
J1> show route aspath-regex ^65502$ terse
A Destination        Next hop      AS path
*172.16.2.0/24       > to 10.0.21.1   65502 i
```

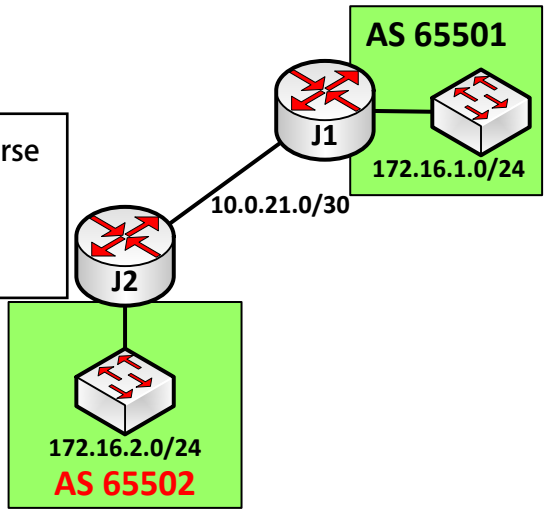


Figure 5.23

Explanation:

- The search was done on the IP routing table because the "show command" did not specify that the protocol was BGP.
- JUNOS has the flexibility to search the routing table using the source protocol as a reference. For example:

```
felix@J1> show route protocol bgp aspath-regex ^65502$ terse
A Destination        Next hop      AS path
*172.16.2.0/24       > to 10.0.21.1   65502 i
```

Table 5.14

- It returns exactly the same output as before but the search was specifically done in routing information sourced by the BGP protocol.

- The following statement also returns the same output.

```
felix@J1> show route aspath-regex 65502 terse
A Destination        Next hop       AS path
*172.16.2.0/24       > to 10.0.21.1   65502 i
```

Table 5.15

Statement: show the routes that have AS65503 in the path.

```
felix@J1> show route protocol bgp aspath-regex .65503. terse
A Destination        Next hop       AS path
*172.16.4.0/24       > to 10.0.21.1   65502 65503 65504 i
```

Table 5.16

Explanation:

- The periods are wild cards that take the place of any AS number.
- Basically the expression .65503. can be read as "anything 65503 anything".

Statement: in J2, show **the path** that only has 65503 and 65504 in that order.

```
felix@J2> show route protocol bgp aspath-regex "65503 65504" terse
A Destination        Next hop       AS path
*172.16.4.0/24       > 10.0.32.1    65503 65504 i
```

Table 5.17

Explanation:

- Notice that the regular expression is delimited by quotation marks. JUNOS requires quotation marks when the regular expressions have more than one AS terms.

Statement: in J2, show **any paths** that have 65503 and 65504 in this order.

```
felix@J2> show route protocol bgp aspath-regex ".* 65503 65504.*" terse
A Destination        Next hop     AS path
*172.16.4.0/24     > 10.0.32.1    65503 65504 i
```

Table 5.18

Explanation:

- The symbols **.*** together mean "any combination as many times" so any path that has 65503 and 65504 in that order will have a match.
- Notice that the regular expression is delimited by quotation marks. JUNOS requires quotation marks when the regular expressions have more than one AS terms and also when the regular expression contains operator symbols.

Statement: in J4, show **any path** that has 65503 and 65501 in this order but with **any ASN in between**.

```
felix@J4> show route protocol bgp aspath-regex "65503 . 65501" terse
A Destination        Next hop    AS path
* 172.16.1.0/24    >10.0.43.2    65503 65502 65501 i
```

Table 5.19

Explanation:

- The dot operator means any other AS term so as long as the expression string has the format 65503 ASN 65501 it will be matched.

Statement: in J4, show **any path** that has 65503 and 65501 in this order but with **any ASN in between**.

```
felix@J4> show route protocol bgp aspath-regex "65503 .* 65501" terse
A Destination        Next hop      AS path
* 172.16.1.0/24      >10.0.43.2    65503 65502 65501 i
```

Table 5.20

Explanation:

- The dot operator followed by asterisk means others AS are in between 65503 and 65501, not just one ASN. In the case topology, only one patch matches this expression.

Statement: in J4, show **any path** that has 65503 anywhere.

```
felix@J4> show route protocol bgp aspath-regex ".* 65503 .*" terse
A Destination        Next hop      AS path
* 172.16.1.0/24      >10.0.43.2    65503 65502 65501 i
* 172.16.2.0/24      >10.0.43.2    65503 65502 i
* 172.16.3.0/24      >10.0.43.2    65503 i
```

Table 5.21

Explanation:

- The dot operator followed by asterisk means any other ASN before or after 65503 including none.

Implementing BGP Route Filtering

The following network will be used to demonstrate the principles of BGP route filtering. These are the general settings:

- The policies will be implemented: in R2 outbound to R3 or in R3 inbound from R2.
- The effect of the policies will be observed in router R3.
- Autonomous system AS1 contains eight IP route prefixes for advertisement by R1.
- Router R1 will be sourcing these IP prefixes toward AS2.
- These prefixes have been selected to have a variety of mask lengths from /17 to /24.

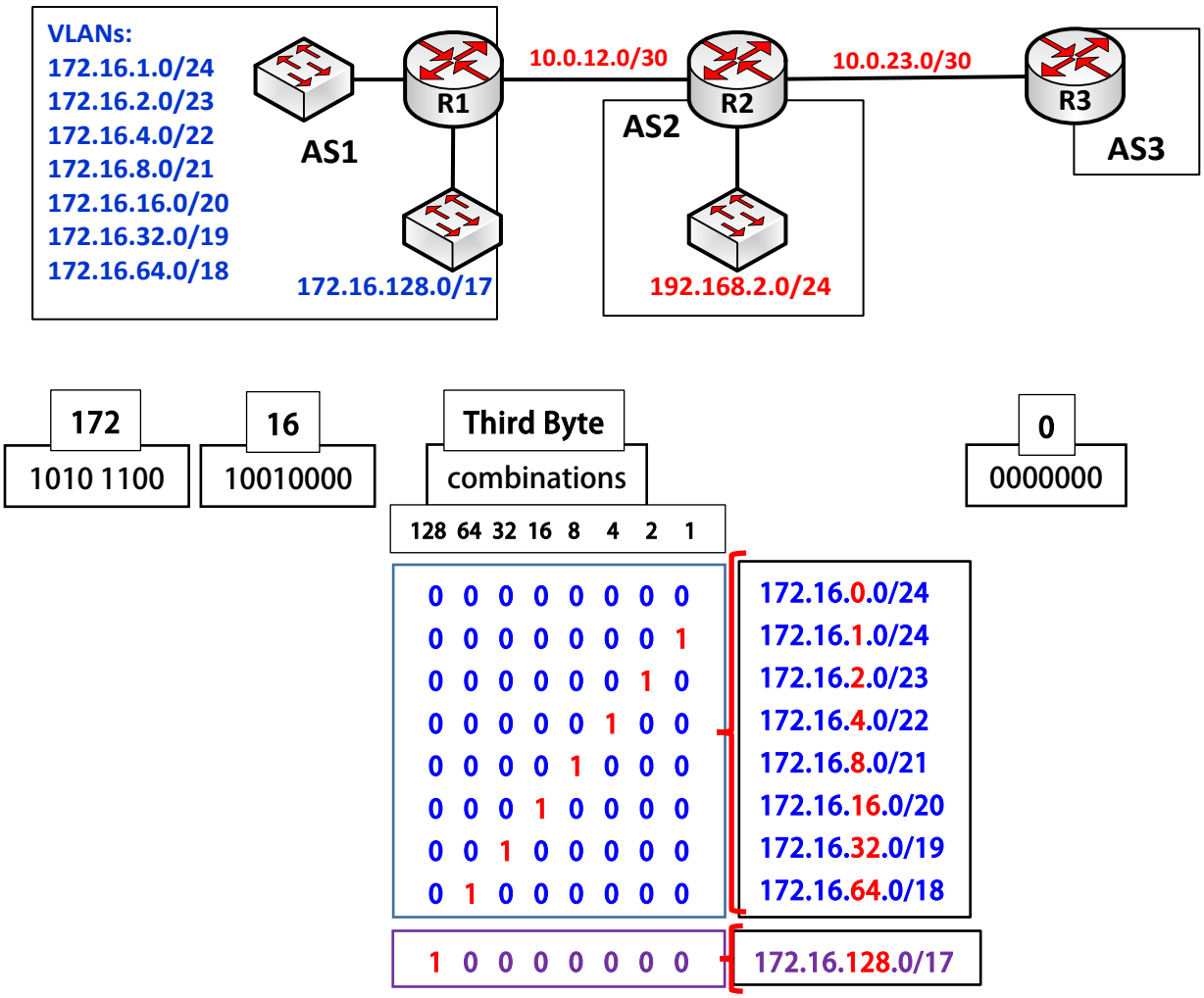


Figure 5.25

These are the basic BGP configurations of these routers:

```
R1#
router bgp 1
network 172.16.1.0 mask 255.255.255.0
network 172.16.2.0 mask 255.255.254.0
network 172.16.4.0 mask 255.255.252.0
network 172.16.8.0 mask 255.255.248.0
network 172.16.16.0 mask 255.255.240.0
network 172.16.32.0 mask 255.255.224.0
network 172.16.64.0 mask 255.255.192.0
network 172.16.128.0 mask 255.255.128.0
neighbor 10.0.12.2 remote-as 2
```

```
R2#
router bgp 2
network 192.168.2.0
neighbor 10.0.12.1 remote-as 1
neighbor 10.0.23.2 remote-as 3
```

```
R3#
router bgp 3
neighbor 10.0.23.1 remote-as 2
```

Table 5.22

Cisco IOS Distribute-list

The distribute list is a tool available in Cisco IOS to filter IP prefixes. The distribute list is specified under the BGP configuration and it is applied to the BGP neighbours with the "neighbor" command. If it is applied outbound then the BGP UPDATES will carry only the IP prefixes that pass thru the filter. If the distribute list is applied inbound then the prefixes in the BGP UPDATE will be evaluated against the conditions of the lists and the ones that meet the conditions will be accepted. This is the general format of a Cisco IOS distribute-list:

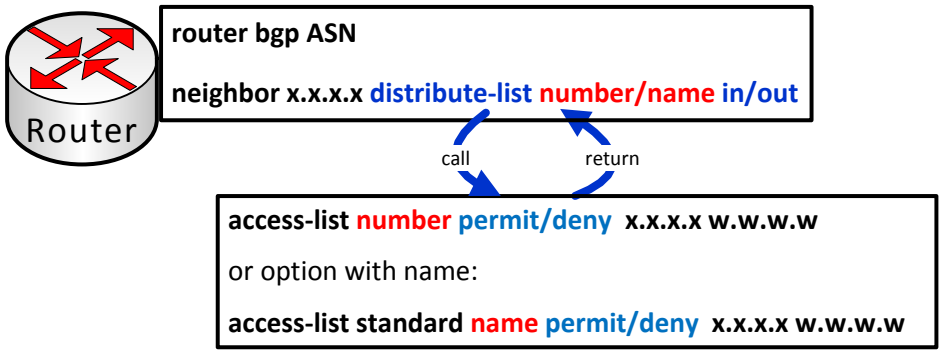


Figure 5.26

The distribute-list makes a call to an access-list where filtering conditions of permitting or denying IP prefixes are specified. Once a match is achieved, the access-list returns the value to the distribute-list. Such IP prefix match is processed by BGP. In short, a distribute-list is a way to implement access-lists under the BGP configuration. Let's see some examples:

Statement: R2-AS2 advertises only prefix 172.16.1.0/24 to R3-AS3. Anything else is filtered out.

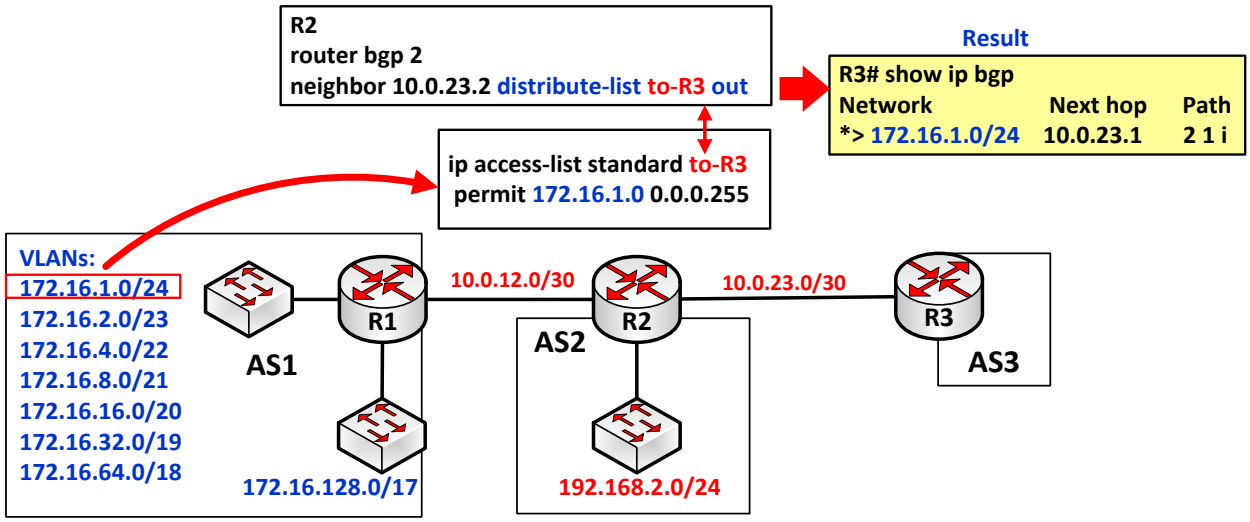


Figure 5.16

Explanation:

- An access-list to-R3 is written in standard format so a name can be given instead of a number.
- The access-list to-R3 matches and permits exactly 172.16.1.0/24 and nothing else.
- There is a "deny implicit" after the permit statement for anything else.
- The distribute-list makes a call to the access-list which filters the prefixes, the resulting match is returned to the distribute-list process, and then BGP prepares the UPDATE message that goes to R3.
- Router R3 only receives the prefix 172.16.1.0/24 and nothing else.

Statement: R2 advertises the prefixes 172.16.1.0/24, 172.16.2.0/23, and 172.16.4.0/22 to R3. Everything else is filtered out.

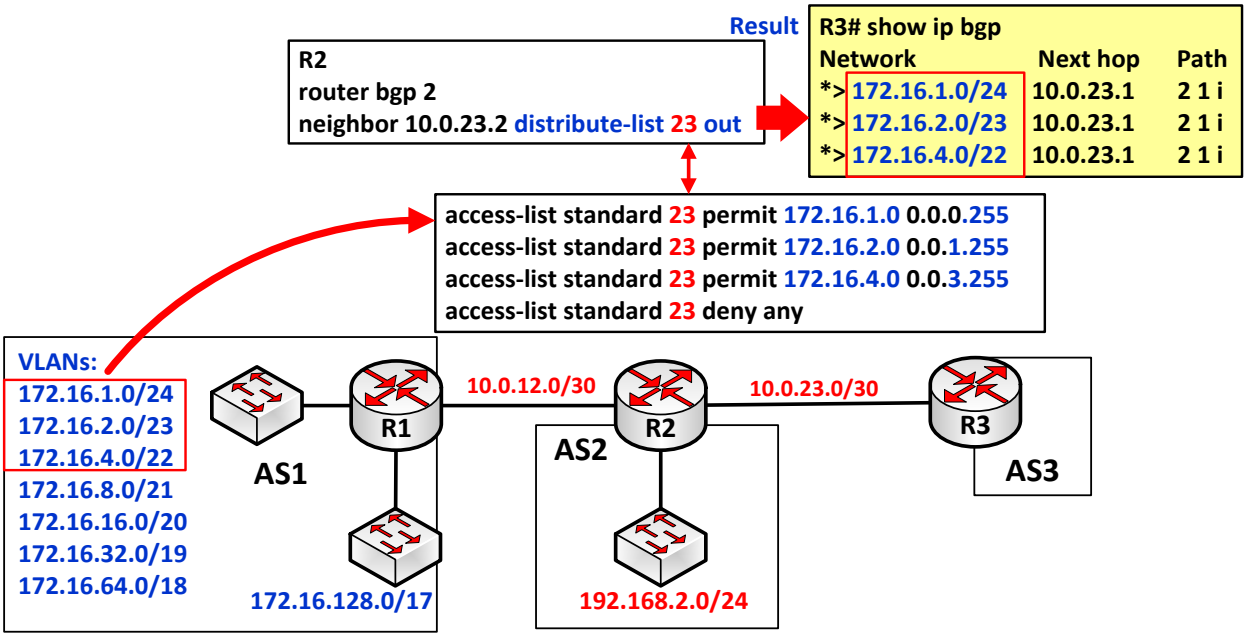


Figure 5.17

Explanation:

- Each statement in the access-list 23 permits a specific prefix.
- The "deny any" at the end is not really necessary since there is an implicit deny in all access-lists.
- A single statement 172.16.0.0 0.0.7.255 would also catch the desired prefixes. However, it would catch other prefixes as well such as 172.16.0.0, 172.16.3.0, 172.16.5.0, and 172.16.7.0 and that might not be the desired effect.
- For example, let's suppose that router R1 would peer with another AS that has the prefixes 172.16.0.0, 172.16.3.0, 172.16.5.0, and 172.16.7.0 and AS3 does not want those. By default, and by mistake, AS3 will receive the prefixes with the policy outline above.
- It is best practice to be more specific than less specific since a filter that is left open can yield unintended consequences.

Cisco IOS Filter-list

As the name indicates, a filter-list sifts thru a sequence of permit or deny conditions matching autonomous systems numbers. The natural way to match ASN is by the use of regular expressions.

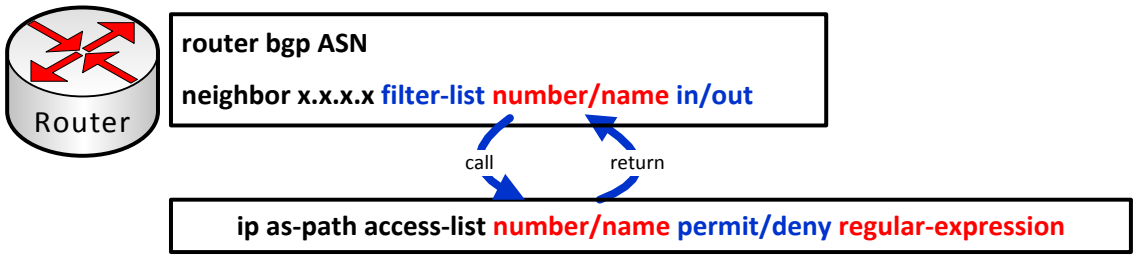


Figure 5.18

Statement: R2-AS2 advertises to R3-AS3 only the internal routes (of AS2) and nothing else.

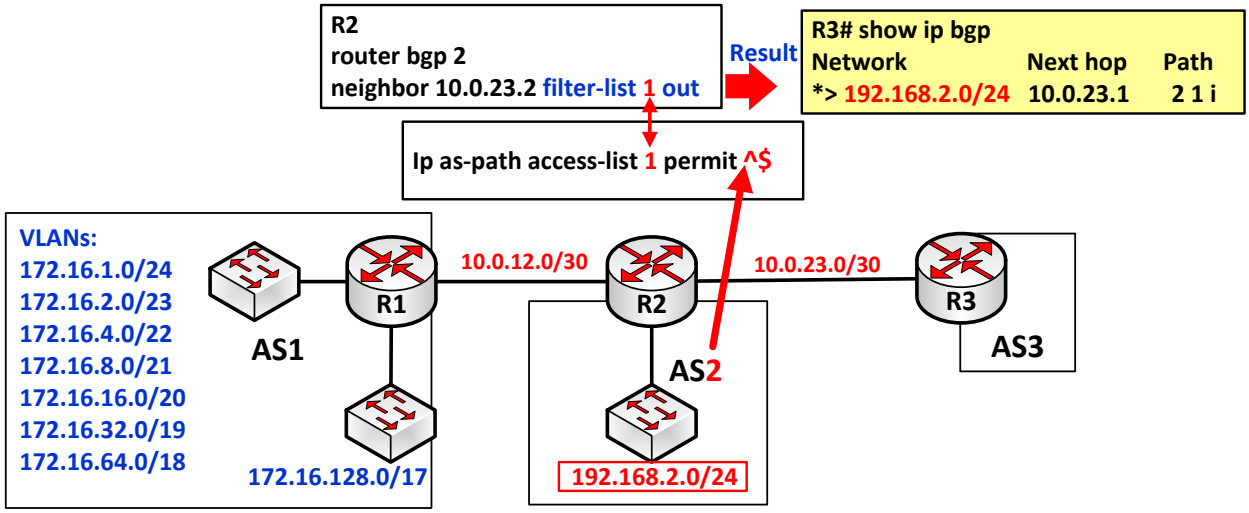


Figure 5.19

Explanation:

- The filter-list makes a call to the ip as-path access-list number 1.
- This ip as-path is an access-list for autonomous system numbers.
- The match is done with the "**empty path**" ^$ regular expression.
- Another way to achieve the same would be to match the prefix with a distribute-list.

Statement: R3-AS3 only allows routes sourced in AS1 and it blocks everything else.

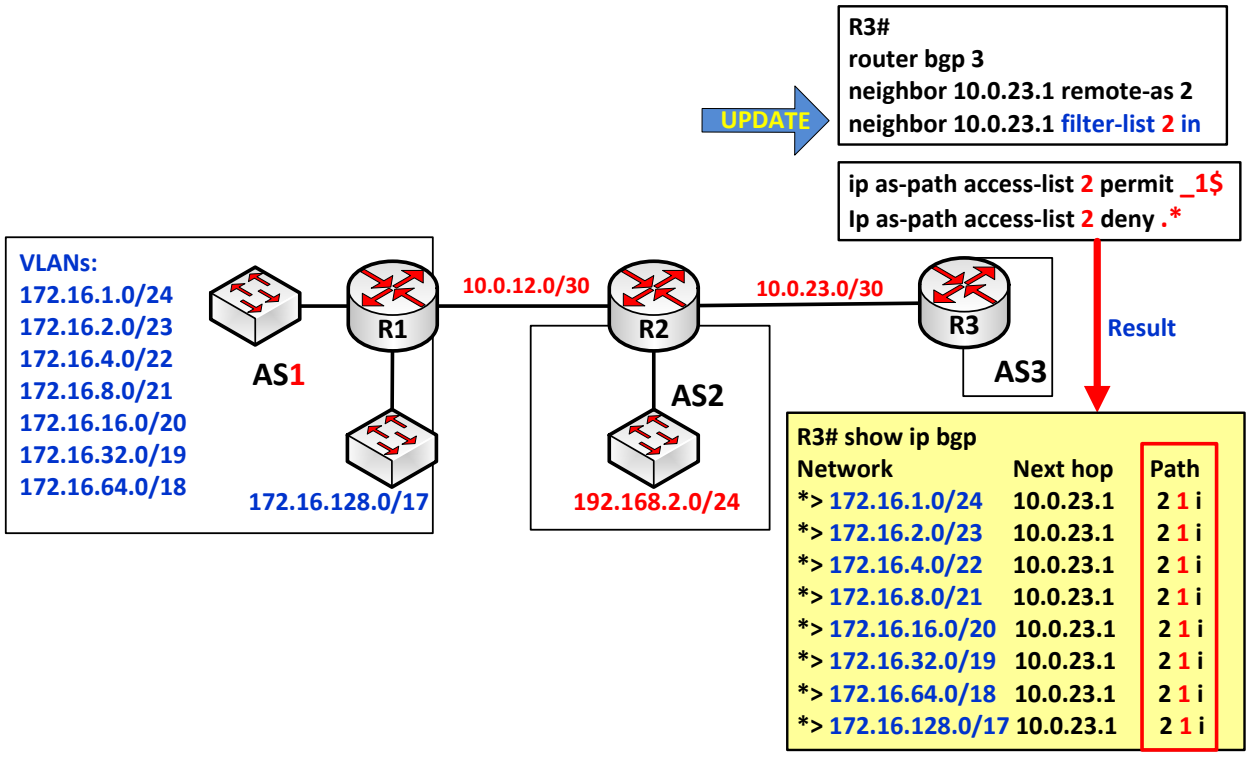


Figure 5.20

Explanation:

- R3 applies a filter-list inbound to filter out everything except the routes sourced in AS1.
- The match is done with the regular expression **_1$** which means "sourced in AS1 with any path after that".
- AS2 route 192.168.2.0/24 is not reachable from AS3 because the filter denies with the regular expression **.*** which means "all AS numbers". This statement to deny is not really needed since there is an implicit deny at the end of the access-list.
- The resulting overall effect is that AS2 becomes a transit AS between AS1 and AS3.

Cisco IOS Prefix-list

The Prefix-list is a sequence of statements to permit or deny IP addresses with their masks. The prefix-list is applied inbound or outbound and it acts as a filter either allowing or blocking IP prefixes depending of the case. The statements are executed in sequence. The prefix-list can be either named or numbered. This is the syntax of the prefix-list:

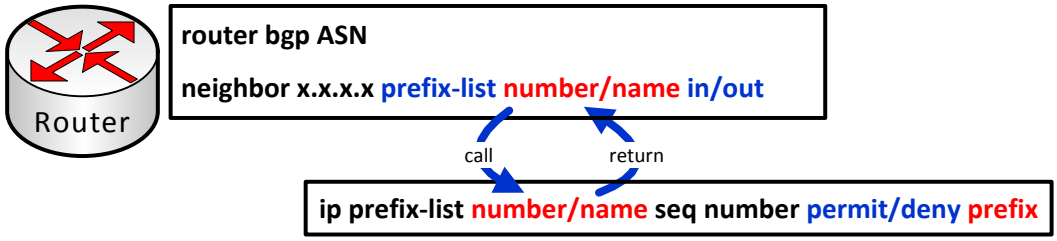


Figure 5.21

It is a common practice to start the sequence number in 5, then jump to 10, and so on. This system leaves room for future additions or changes to the prefix-list.

Statement: permit only the prefixes 172.16.1.0/24, 172.16.2.0/23, and 172.16.4.0/22.

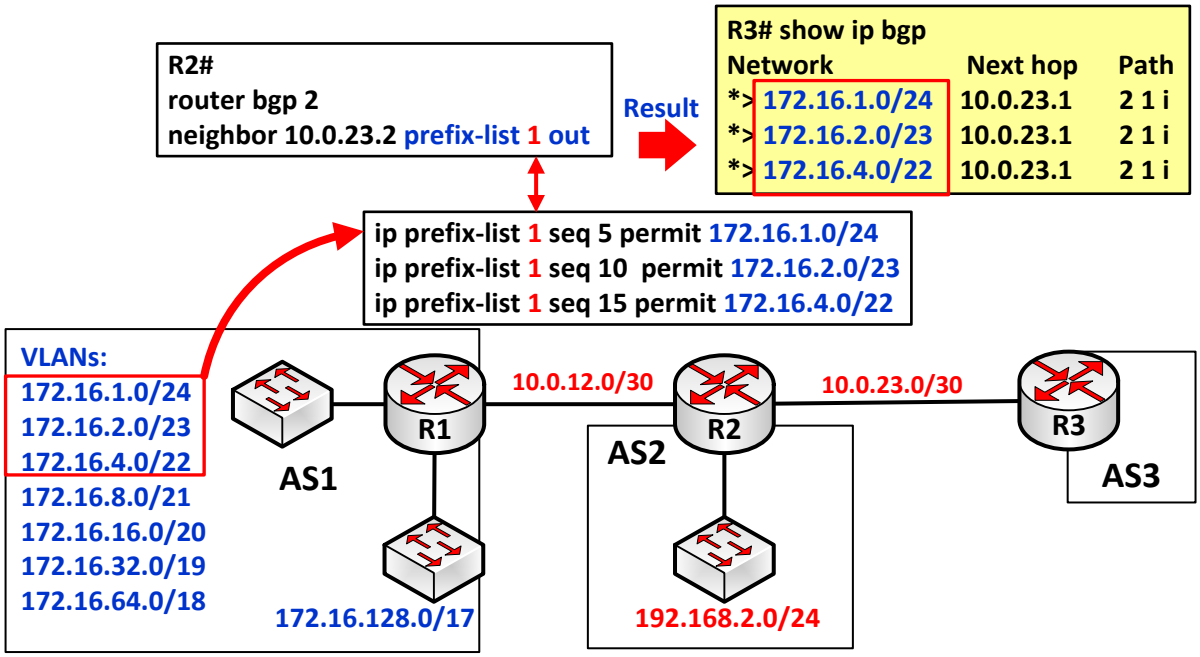


Figure 5.22

Explanation:

- Each sequence allows the exact prefix only.
- There is an "implicit deny" at the end of the prefix-list.

Statement: permit only 172.16.8.0/21, 172.16.16.0/20, and 172.16.32.0/19.

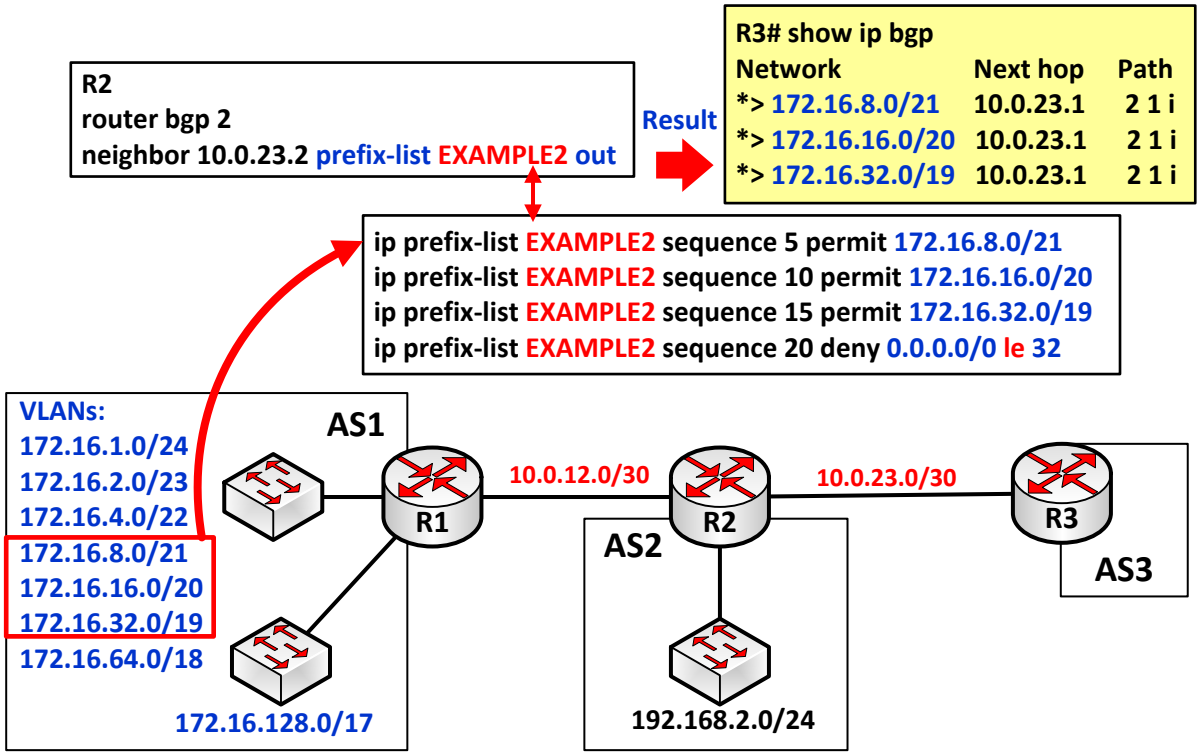


Figure 5.23

Explanation:

- First, the three desired prefixes are allowed to pass thru.
- Then an explicit deny has been added at the end of the statements. It is really not necessary since there would be an implicit deny anywhere, but it is there for the sake of demonstration.
- The expression 0.0.0.0/0 le 32 means catch all the bits from bit 0 to bit 32; hence it catches all of the IPv4 addresses.

Statement: repeat the previous action, but in this case, catch the three prefixes with only one statement if possible.

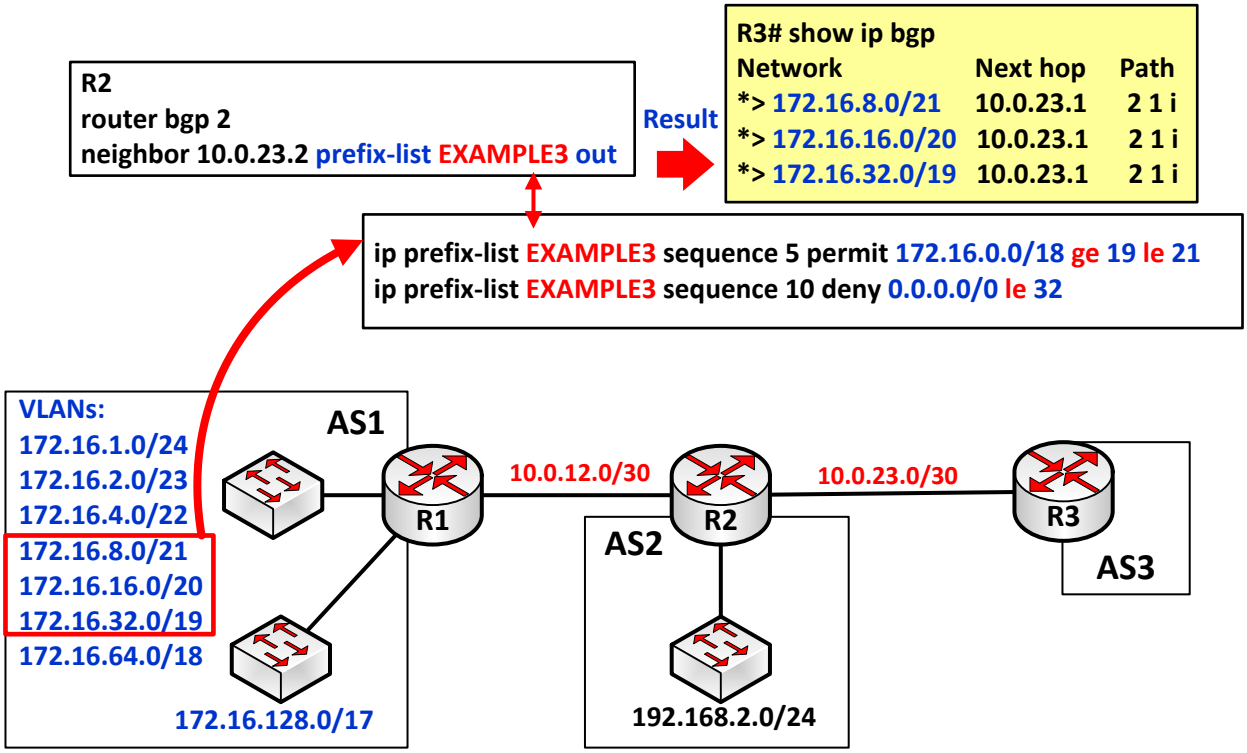


Figure 5.24

Explanation:

- The three IP routes share a common prefix (172.16.) for the first two bytes.
- Their difference is in the third byte in the specific bits worth 32, 16, and 8:

Bit 17	Bit 18	Bit 19	Bit 20	Bit 21	Bit 22	Bit 23	Bit 24
128	64	32	16	8	4	2	1
0	0	0	0	1	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0

Table 5.23

- The expression 172.16.0/18 ensures that the first eighteen bits match exactly.
- The expression ge 19 (greater than 19) means start catching the bits from position 19.
- The expression le 21 (less than 21) means stop considering the bits after bit 21.

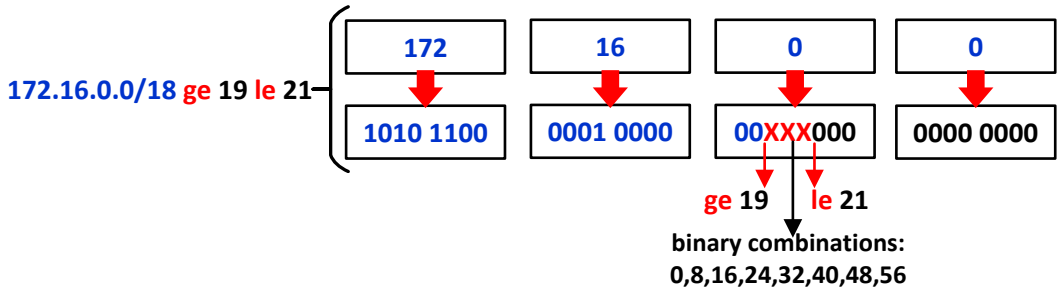


Figure 5.25

- The prefix-list expression catches the third byte combinations 32, 16, and 8 which is the intention of the filter.
- However, the expression would also catch other combinations such as 0, 24, 40, 48, and 56 in the case that they were present in the routing table of R1 and that is not necessarily the intention of the policy.
- Even though, this filter is apparently working, this implementation is too loose and it would not be considered best practice.
- So in this particular situation, it is better to filter the prefixes as it was done in EXAMPLE2.

Statement: catch all the prefixes that used to belong to a Class B address, but the prefixes must have a mask longer or equal than /21 and shorter or equal than 23.

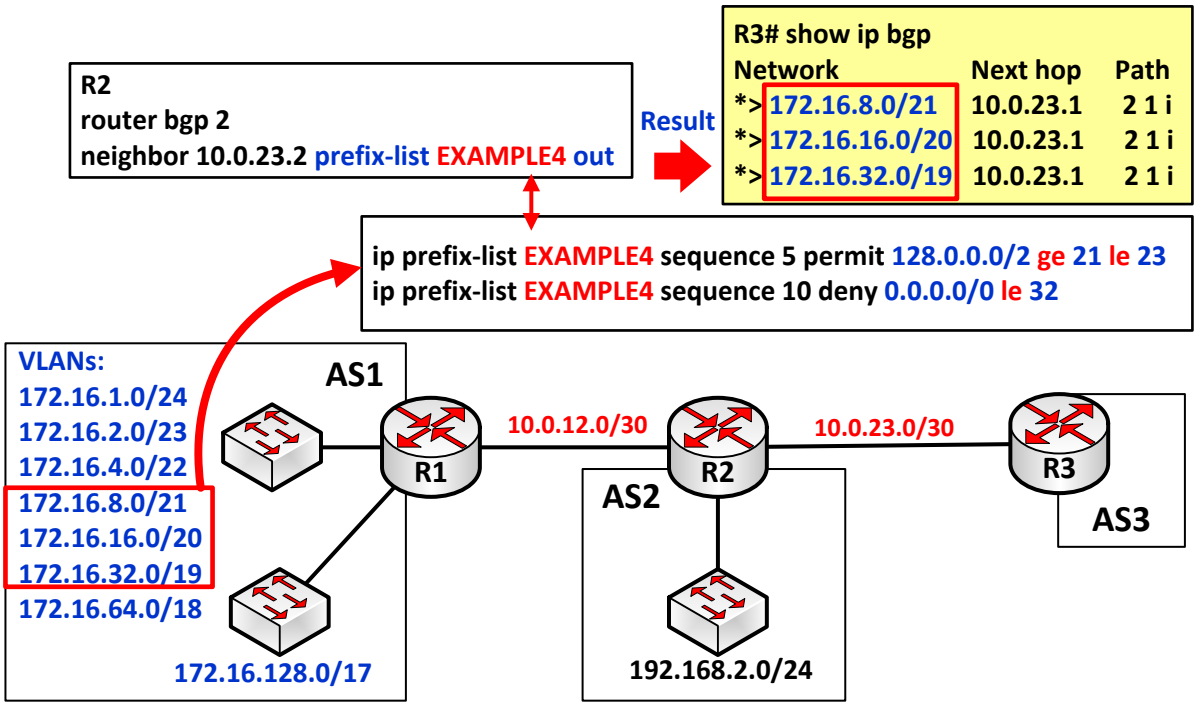
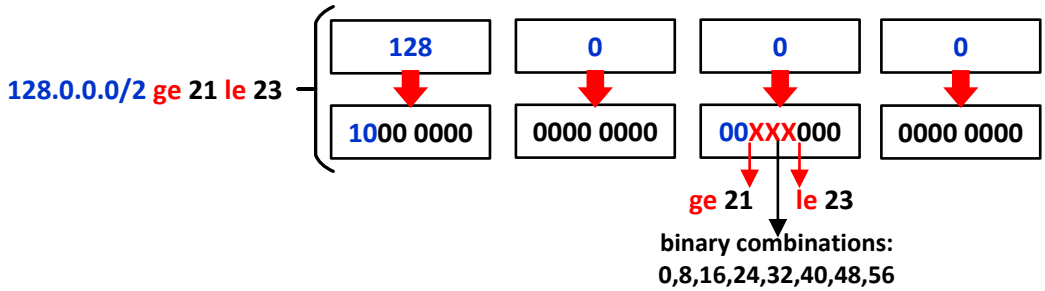


Figure 5.26

Explanation:

- The prefix-list allows IP prefixes that begin with the binary pattern 10 in the first byte or 128 in decimal. $(10xxxxxx)_2$
- The IP addresses that begin with 172 have such pattern, so all the 172.16 routes have a match.
- Furthermore, the ge and le operators constrain the routes to the ones that have masks within 21 and 23 bits long.
- The following diagram illustrate this situation:



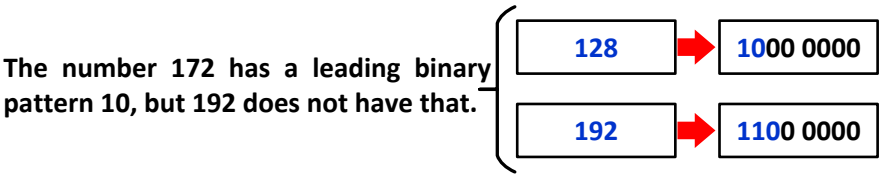


Figure 5.27

Cisco IOS Route-map

Cisco Route-maps are frameworks to implement policies by integrating the use of the other tools. For example, a route-map can make a call to an access-list or to a prefix-list to match an address and then execute an action or set a condition. Route-maps are very versatile and they widely used in BGP policies. This is the general structure of the route-map:

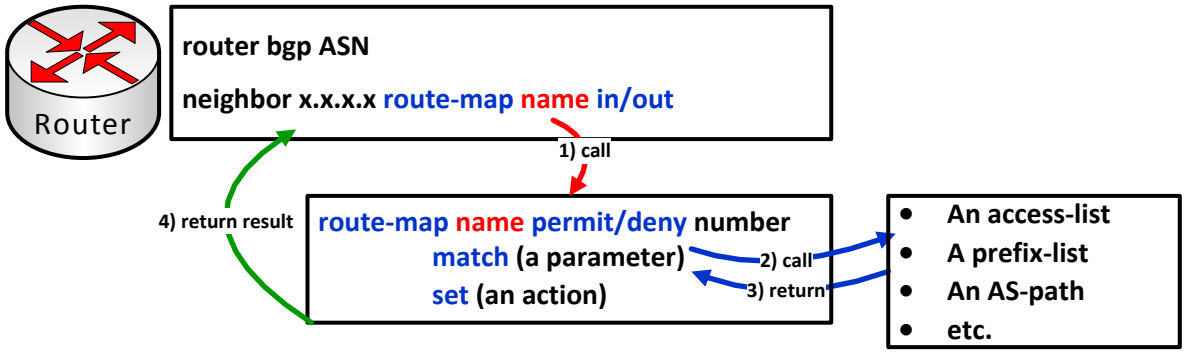


Figure 5.28

Route-maps have a wide range of options available. The main body of the route-map contains statements that permit or deny an action or a condition. The first role of each statement is to match a parameter for example an ip route or a BGP attribute.

R2 (config)#**route-map EXAMPLE permit 10**

R2 (config-route-map)#match ?

as-path	Match BGP AS path list
community	Match BGP community list
extcommunity	Match BGP/VPN extended community list
interface	Match first hop interface of route
ip	IP specific information
ipv6	IPv6 specific information
length	Packet length
local-preference	Local preference (BGP LOCAL-PREF) for route
metric	Match metric (BGP MED) of route
mpls-label	Match routes which have MPLS labels
nlri	BGP NLRI type
policy-list	Match IP policy list
route-type	Match route-type of route
source-protocol	Match source-protocol of route
tag	Match tag of route

Table 5.24

Let's explore the option "match ip":

```
R2(config)#route-map EXAMPLE permit 10
R2(config-route-map)#match ip ?
  address        Match address of route or match packet
  next-hop       Match next-hop address of route
  route-source   Match advertising source address of route
```

Table 5.25

Further, let's get deeper into "match ip address":

```
R2(config-route-map)#match ip address ?
  <1-199>        IP access-list number
  <1300-2699>    IP access-list number (expanded range)
  WORD           IP access-list name
  prefix-list    Match entries of prefix-lists
```

Table 5.26

The statement "match ip address" links to access-lists and prefix-lists depending of what is being used. This works similar to a sub-routine in programming languages, the main body of the route-map places a call to one of the options listed here. IP route prefixes are matched and the result is sent back to the main body of the route-map to continue with another step.

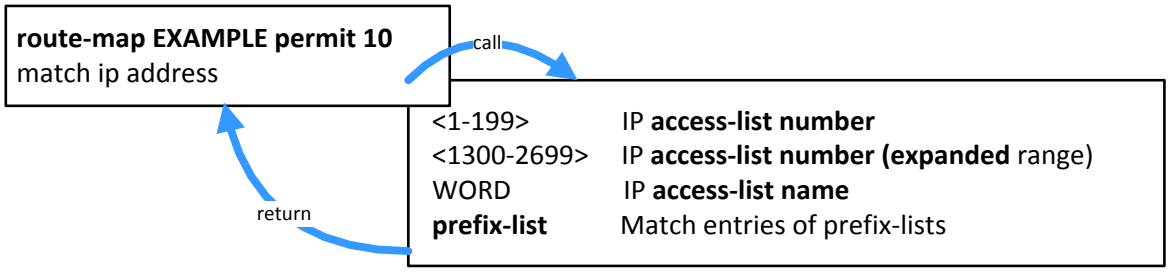


Figure 5.29

Specifically, a route-map statement that permits all the prefixes that begin with 172.16. could make a call to an existing prefix-list:

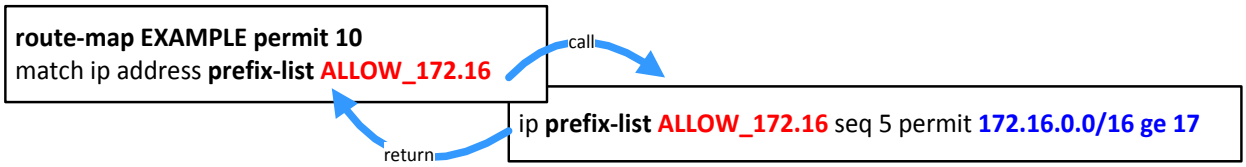


Figure 5.30

Once the prefixes are matched, the next step could be a "set" action and the following list provides the most helpful choices including some of BGP attributes.

R2(config)#**route-map EXAMPLE permit 10**
R2(config-route-map)#**set ?**

as-path	Prepend string for BGP AS-PATH attribute
community	BGP community attribute
comm-list	BGP community list (for deletion)
dampening	BGP route flap dampening parameters
extcomm-list	BGP/VPN extended community list (for deletion)
extcommunity	BGP extended community attribute
local-preference	BGP LOCAL-PREF attribute
nlri	BGP NLRI type
origin	BGP ORIGIN code attribute
weight	BGP Cisco weight
metric	BGP MED attribute. Metric value.
metric-type	Type of metric for destination routing protocol
interface	Output interface
tag	Tag value for destination routing protocol
default	Set default information
ip	IP specific information

Table 5.27

For instance, let's change the metric or MED attribute to 123:

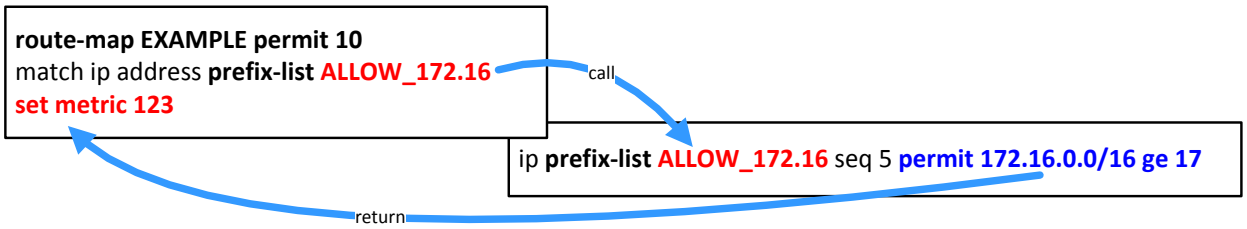


Figure 5.31

The route-map must now be applied toward a neighbour either inbound or outbound. This is the result in the example network:

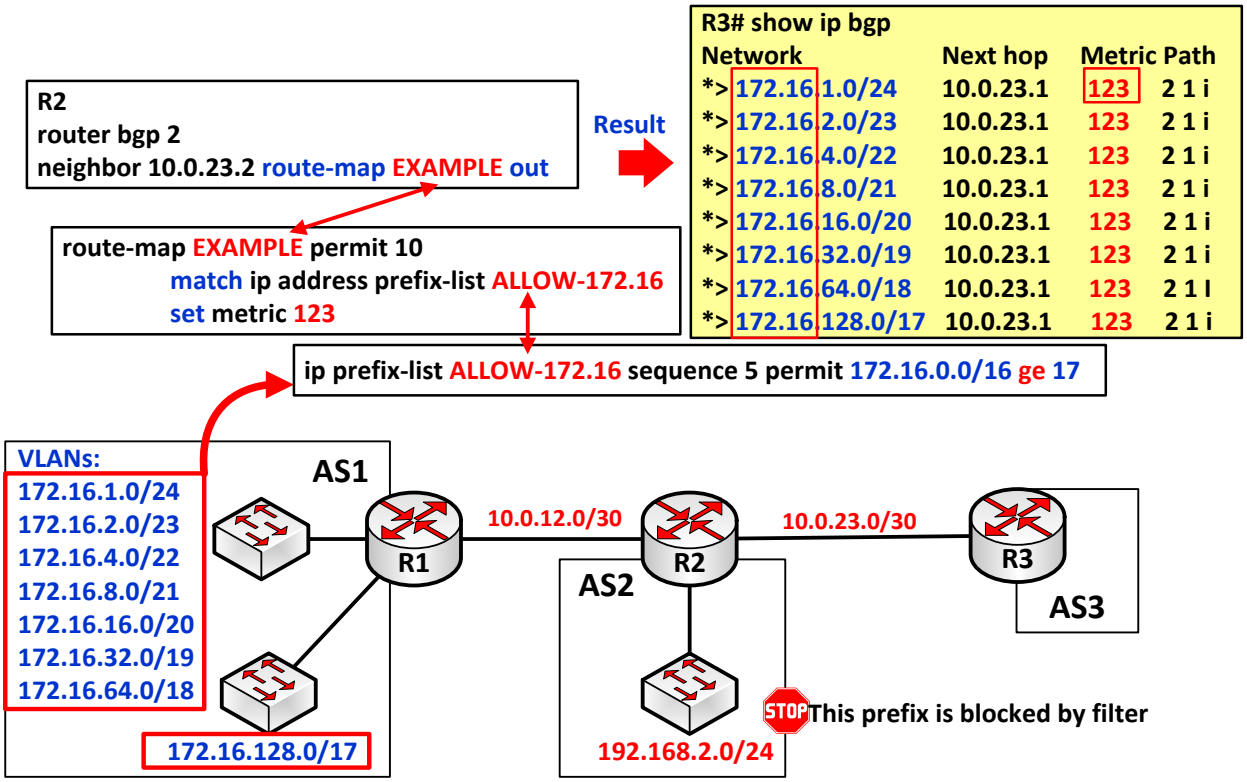


Figure 5.32

All the prefixes that begin with 172.16 are advertised to AS3 with a metric of 123. An additional result of this policy is that all other prefixes are automatically blocked, for example 192.168.2.0/24 does not get advertised to R3. If 192.168.2.0/24 were to be advertised without changing the metric, then the following options are possible:

Option a) A default permit statement allows everything that has not been matched by previous statements. **route-map EXAMPLE permit 20**	**Option b)** A very specific statement to allow only the desired prefix. **route-map EXAMPLE permit 20** **match ip address 2** **access-list 2 permit 192.168.2.0 0.0.0.255**

Statement: implement a policy with a route map that matches all the prefixes advertised by AS1, then set the metric to 123. Deny everything else. (This is the same policy as the case before, but the difference is that the match will be done with the AS number instead of the prefixes).

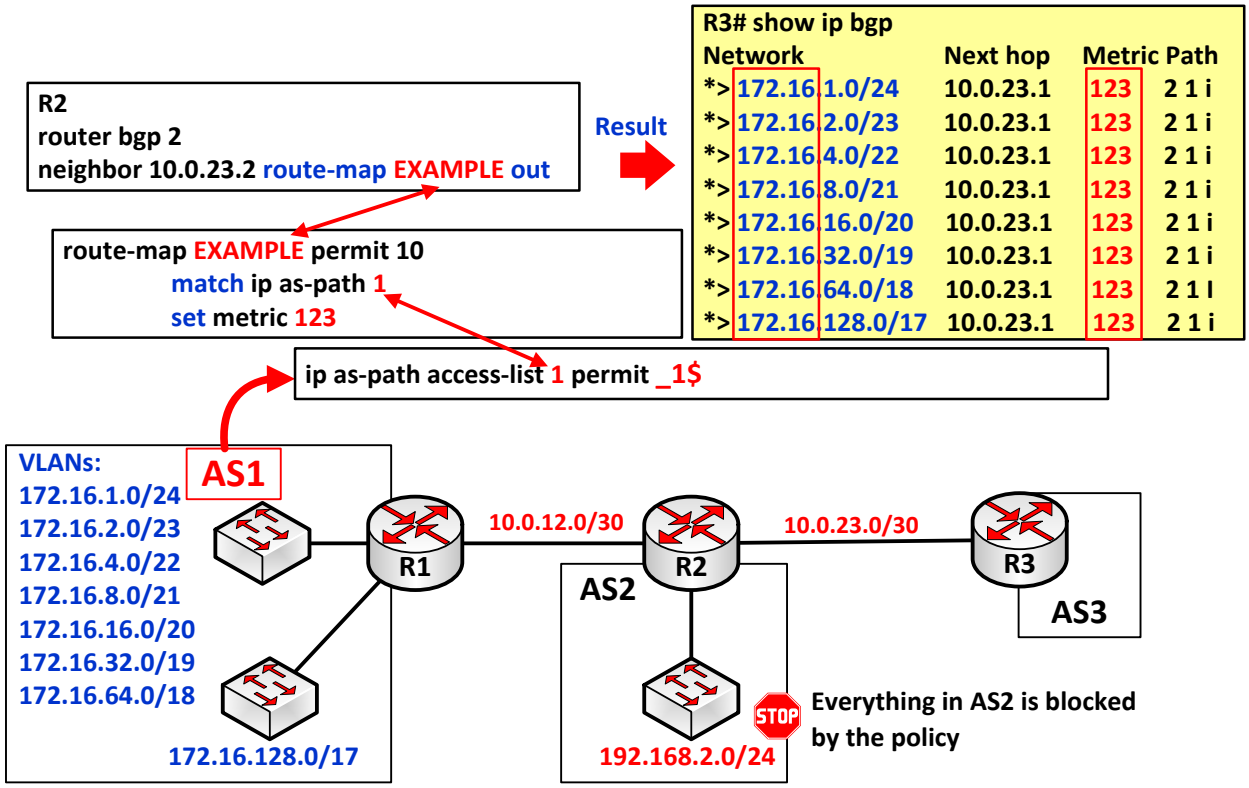


Figure 5.33

Explanation:

- The route-map EXAMPLE2 main body permits a call to match all the routes sourced by autonomous system number one. The filter "as-path access-list #" does that with the regular expression _1$. This effectively matches everything that is sourced by AS1.
- The filter returns all the matched IP prefixes to the main body of the route-map which then it proceeds to set the metric value to 123.
- There are no more statements in the route-map so that implies a deny condition to everything else. This blocks the prefix 192.168.2.0/24 from being advertised to AS3.

Statement: write a policy with a route-map that:

- Set the metric (MED attribute as well) to 10 all the IP routes that being with 172.16.0/17. That means that 172.16.128/17 will not be a match here.
- Set the metric to 20 for prefix 172.16.128/17.
- Set a community attribute 2:30 to any prefix sourced by AS2.
- Permit everything not matched above to be advertised without any change.
- This policy must include one access-list, one prefix-list, and one ip as-path access-list.

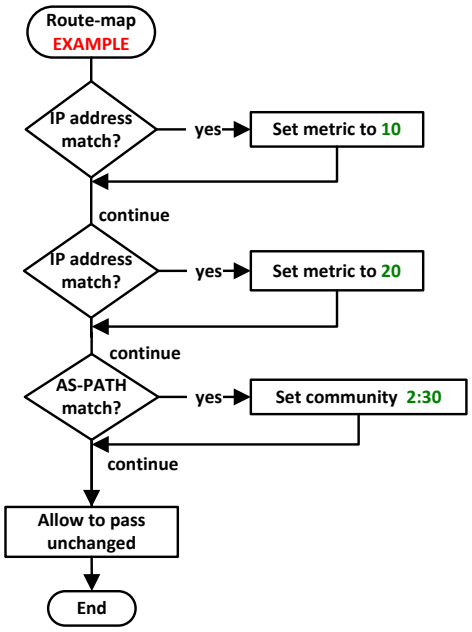
This is the complete policy EXAMPLE:

```
router bgp 2
 network 192.168.2.0
 neighbor 10.0.12.1 remote-as 1
 neighbor 10.0.23.2 remote-as 3
 neighbor 10.0.23.2 route-map EXAMPLE out
!
ip bgp-community new-format
!
route-map EXAMPLE permit 10
 match ip address 10
 set metric 10
!
route-map EXAMPLE permit 20
 match ip address prefix-list 20
 set metric 20
!
route-map EXAMPLE permit 30
 match as-path 30
 set community 2:30
!
route-map EXAMPLE permit 40
!
```

```
access-list 10 permit 172.16.0.0 0.0.127.255
```

```
prefix-list 20 seq 5 permit 172.16.128.0/17
```

```
ip as-path access-list 30 permit ^2$
```



Result

```
R3# show ip bgp
Network          Next hop    Metric Path
*> 172.16.1.0/24     10.0.23.1     10      2 1 i
*> 172.16.2.0/23     10.0.23.1     10      2 1 i
*> 172.16.4.0/22     10.0.23.1     10      2 1 i
*> 172.16.8.0/21     10.0.23.1     10      2 1 i
*> 172.16.16.0/20    10.0.23.1     10      2 1 i
*> 172.16.32.0/19    10.0.23.1     10      2 1 i
*> 172.16.64.0/18    10.0.23.1     10      2 1 i
*> 172.16.128.0/17   10.0.23.1     20      2 1 i
*> 192.168.2.0/24    10.0.23.1     0         2 i
```

Figure 5.34

Explanation:

- This route-map starts by allowing and matching the prefixes covered by 172.16.0/17 which includes from 172.16.1.0 to 172.16.64.0 in the routing table. This is done with an access-list with an ample wildcard mask (not the best practice, but this is an example.)
- For these prefixes the metric is set to 10.
- Then the route-map accepts 172.16.128.0/17 calling upon a prefix-list.
- The match is exact and the metric is set to 20.
- The route-map proceeds now to the next step which is to match everything sourced by AS2 with a call upon an as-path filter.
- AS2 only advertises 192.168.2.0/24 and a community 2:30 is attached to the prefix when it is updated.
- Everything else that did not match any of the previous conditions will be allowed to pass unchanged by the statement "permit 40".

Juniper JUNOS

A JUNOS policy is built upon policy statements. These statements contain lines of instruction that refer to filtering tools and setting actions. Once a policy is completed it can be used, by calling its name, in pretty much any JUNOS process, not just BGP, for example it can be used by other routing protocols and security. Also, JUNOS syntax is the same across different platforms. This is the most general format of a JUNOS policy statement:

```
JUNOS#
edit policy-options policy-statement EXAMPLE
        set term 1 from "something"
        set term 1 then "do something"
```

Basically, a "container " named EXAMPLE will contain instruction lines that will execute in order. The policy statement has terms that match conditions and then they either accept or reject or execute another action like the modification of an attribute. The terms can be numbered or named, like 1,2,3 or one, two, three, or A, B, C, etc. It is also a common practice that the terms names relate to the action that they execute as in "deny-private-address", or "allow-default-route" for example.

Once the policy statement is defined it can be used and reused by a JUNOS procedure by just making a reference to its name.

JUNOS Route-Filter

This is one of the tools available in JUNOS for filtering routing information. If configured in a policy statement term then it matches route prefixes depending of the following options:

Match	Meaning
exact	Match exactly the prefix length
longer	Match the prefixes with a mask longer than the reference mask
orlonger	Match the prefixes with a mask equal or longer than the reference mask
Prefix-length-range /xx-/zz	Match prefixes between the lower (xx) and upper (zz) boundaries
through	Route lies between two prefixes
upto	Match prefixes up to the indicated mask value

Table 5.28

To explain JUNOS route-filters, the topology from the example network will be use once again but this time all the routers are Juniper. This is the "skeleton" configuration of the network.

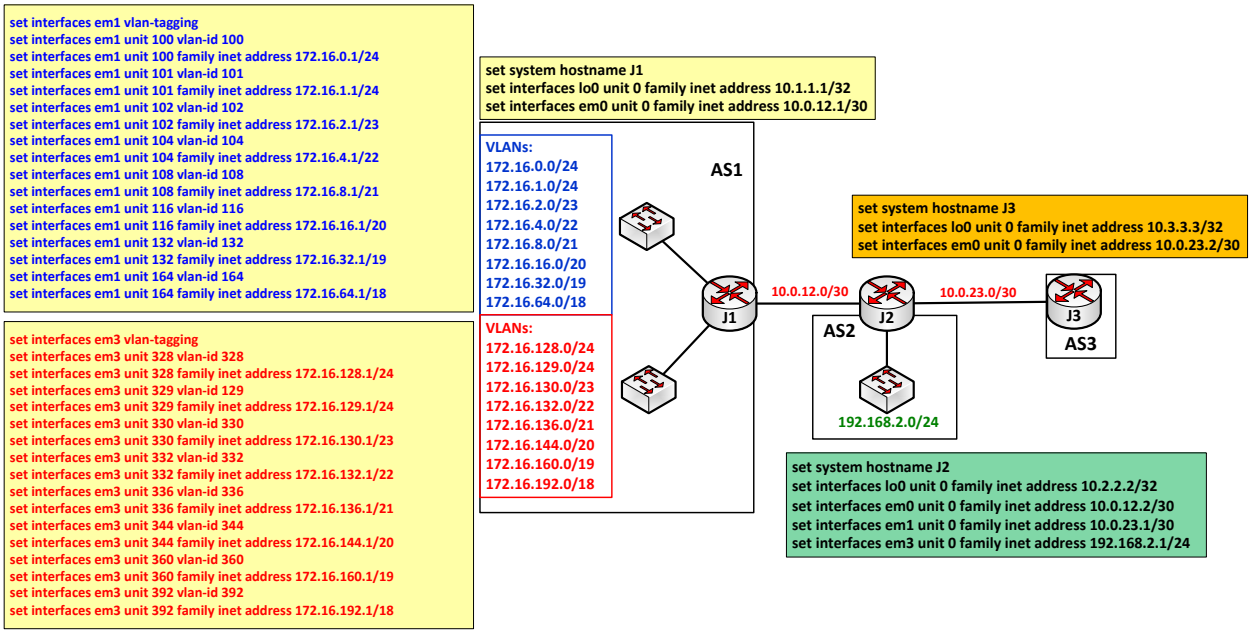


Figure 5.35

Let's begin by stating the routing intentions:

Statement: router J1@AS1 will advertise using BGP all the directly connected routes except the loopback address.

The following is configured in router J1@AS1:

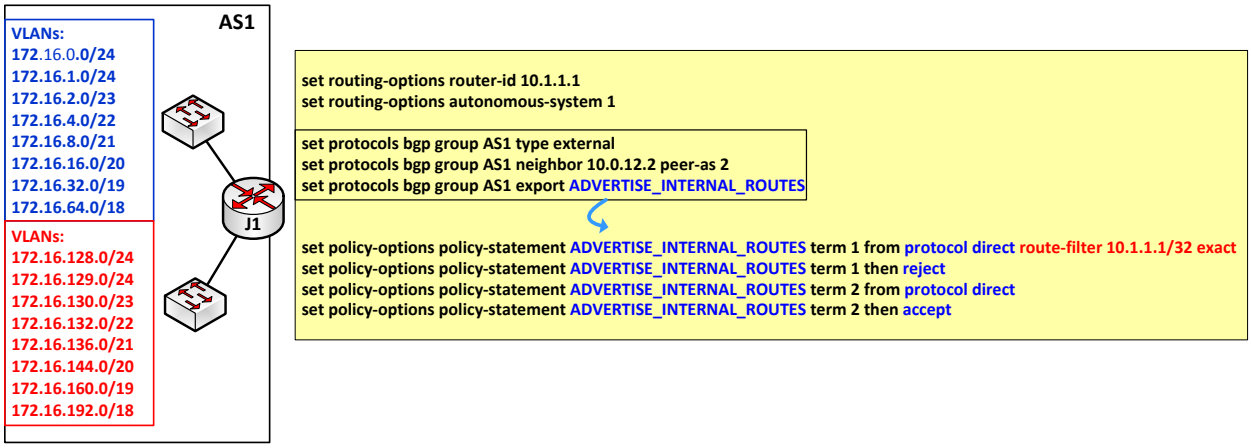


Figure 5.36

The policy ADVERTISE_INTERNAL_ROUTES has two terms, the first term matches exactly prefix 10.1.1.1/32 and rejects it because the administrator does not wish to advertise this loopback address. After that, term 2 accepts everything that is directly connected or all the prefixes from the LAN switches. This achieves the desired export policy, however a better, more detailed, statement term 2 would be:

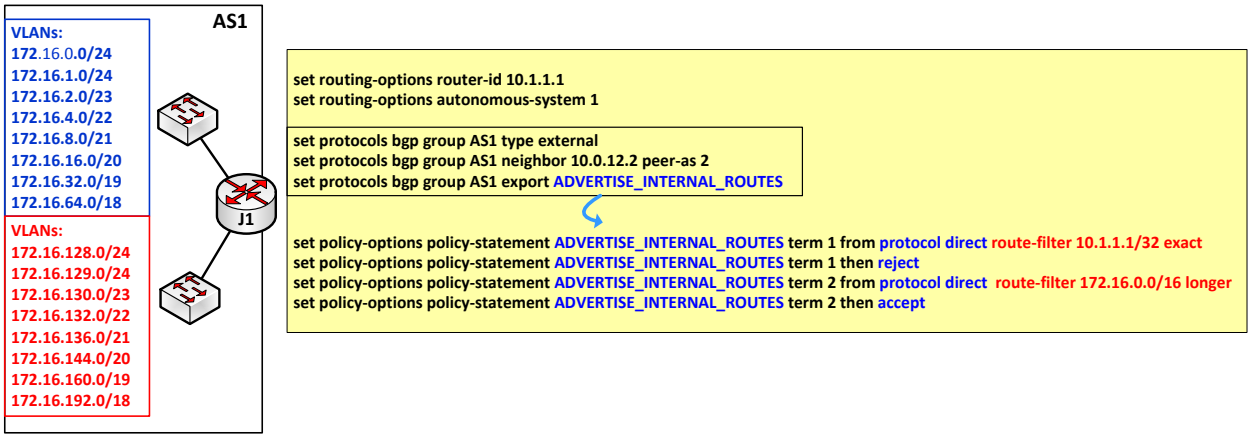


Figure 5.37

The route-filter 172.16.0.0/16 longer catches all the prefixes that begin with 172.16 with a mask longer than 16 (not including 16) and that matches the routing intention of AS1 which is to advertise its internal routes only. The previous policy could be further simplified since the IP prefixes are present in the routing table there is no need to match the "from protocol direct". In such case, the route-filter alone would match the desired prefix.

```
edit policy-options policy-statement ADVERTISE_INTERNAL_ROUTES
        set term 1 from route-filter 10.1.1.1/32 exact
        set term 1 then reject
        set term 2 from route-filter 172.16.0.0/16 longer
        set term 2 then accept
```

Furthermore, what the administrator intends to do is just to advertise the prefixes that start with 172.16. local to AS1, so the following policy also achieves the same result:

```
edit policy-options policy-statement ADVERTISE_INTERNAL_ROUTES
        set term 1 from route-filter 172.16.0.0/16 longer
        set term 1 then accept
        set term 2 then reject
```

The policy closes with a rejection for anything that is not matched by the previous statement. Notice that AS1 does not do transit to any other AS. If that were the case, then another statement would have to be added to allow BGP to propagate information.

The next table contains all the resulting IP prefixes installed in the neighbour router J2. This table will be used for the next exercises to filter routing information.

```
felix@J2> show route protocol bgp terse

A Destination        Next hop     AS path
* 172.16.0.0/24      >10.0.12.1   1 i
* 172.16.1.0/24      >10.0.12.1   1 i
* 172.16.2.0/23      >10.0.12.1   1 i
* 172.16.4.0/22      >10.0.12.1   1 i
* 172.16.8.0/21      >10.0.12.1   1 i
* 172.16.16.0/20     >10.0.12.1   1 i
* 172.16.32.0/19     >10.0.12.1   1 i
* 172.16.64.0/18     >10.0.12.1   1 i
* 172.16.128.0/24    >10.0.12.1   1 i
* 172.16.129.0/24    >10.0.12.1   1 i
* 172.16.130.0/23    >10.0.12.1   1 i
* 172.16.132.0/22    >10.0.12.1   1 i
* 172.16.136.0/21    >10.0.12.1   1 i
* 172.16.144.0/20    >10.0.12.1   1 i
* 172.16.160.0/19    >10.0.12.1   1 i
* 172.16.192.0/18    >10.0.12.1   1 i
```

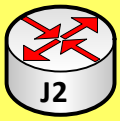


Figure 5.37

Statement: router J2@ AS2 accepts the routes advertised by J1@AS1 and then advertises to J3@AS3 the directly connected 192.168.2.0/24 **only**, **plus** the routes that have the prefix 172.16.0 until bit 20. Deny everything else. In router J2@AS2 the following is configured:

```
set routing-options router-id 10.2.2.2
set routing-options autonomous-system 2

set protocols bgp group AS2 type external
set protocols bgp group AS2 neighbor 10.0.12.1 peer-as 1
set protocols bgp group AS2 neighbor 10.0.23.2 peer-as 3
set protocols bgp group AS2 export EXAMPLE

set policy-options policy-statement EXAMPLE term 1 from route-filter 192.168.2.0/24 exact
set policy-options policy-statement EXAMPLE term 1 then accept
set policy-options policy-statement EXAMPLE term 2 from protocol bgp
set policy-options policy-statement EXAMPLE term 2 from route-filter 172.16.0.0/20 orlonger
set policy-options policy-statement EXAMPLE term 2 then accept
set policy-options policy-statement EXAMPLE term 3 then reject
```

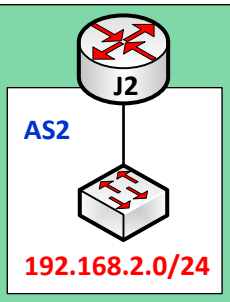


Figure 5.38

Router J2 has one internal prefix to advertise, 192.168.2.0/24, and it does so with a route-filter matching exactly this IP route. The other prefixes to be advertised to J3@AS3 are originated by the BGP updates from J1@AS1, but not all the prefixes only those with masks equal or longer than /20 that have the preceding prefix 172.16.0.0, in binary terms:

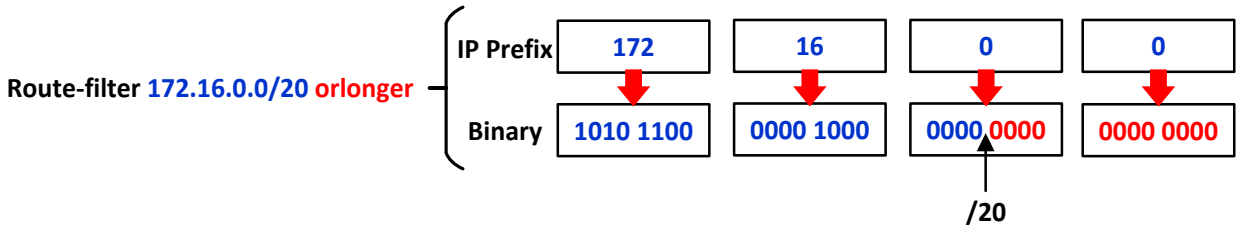


Figure 5.39

This is the resulting table in J3@AS3 with the summaries of the route-filters.

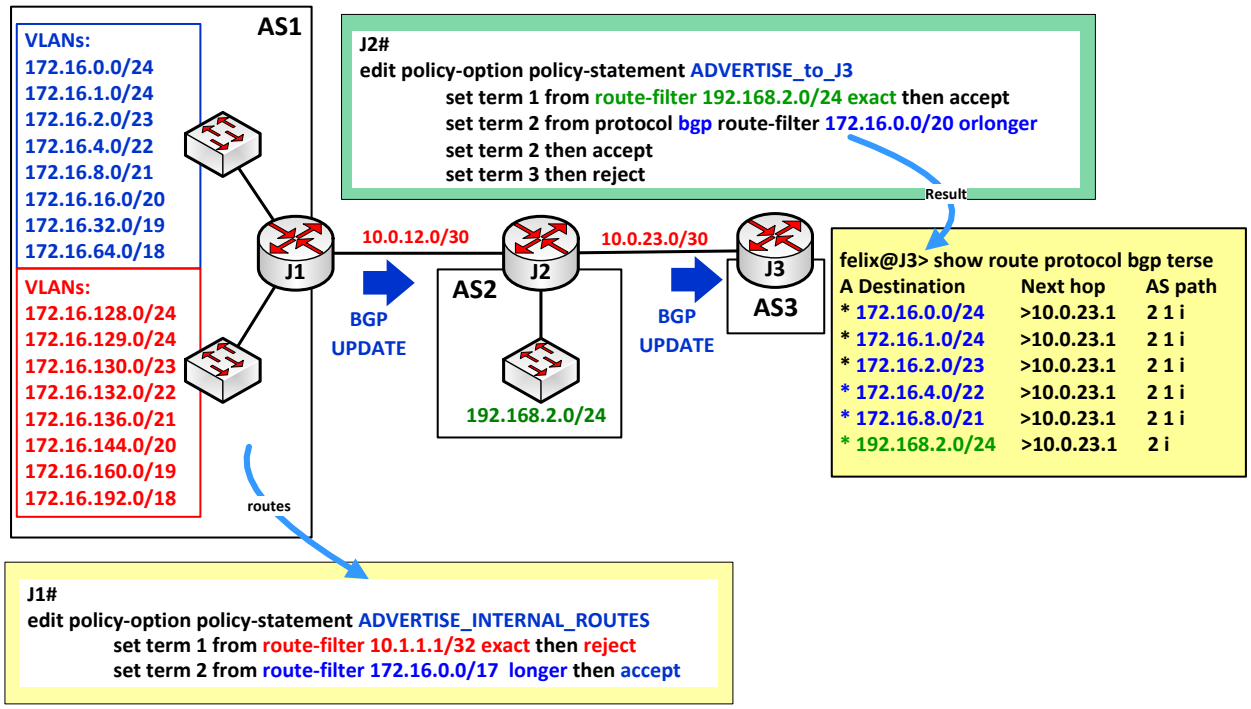


Figure 5.40

- Prefixes 172.16.8.0/21, 172.16.4.0/22, 172.16.2.0/23, 172.16.0.0/24, and 172.16.1.0/24 have masks equal or longer than /20, so this is correct.
- But notice that other prefixes such as 172.16.16.0/20, 172.16.32.0/19 and 172.16.128.0/24 also have masks equal or longer than /20; nevertheless they are not caught by the route-filter. The expression 172.16.0/20 means that the first 20 bits of the

prefix have to match. Specifically, the third byte of the addresses has to begin with 0000 and that excludes the values 16, 32, 64, 128,130, etc.

Statement: in J2@AS2 advertise to J3@AS3 the prefix 192.168.2.0/24 and the prefixes that belong to the range 172.16.0.0/17 up to /18 only.

The previous example configuration does not need to be altered only the term 2 of the policy-statement EXAMPLE needs to be edited:

```
edit policy-options policy-statement EXAMPLE term 2
        set from protocol bgp route-filter 172.16.0.0/17 upto /18
        set then accept
```

This results in the following:

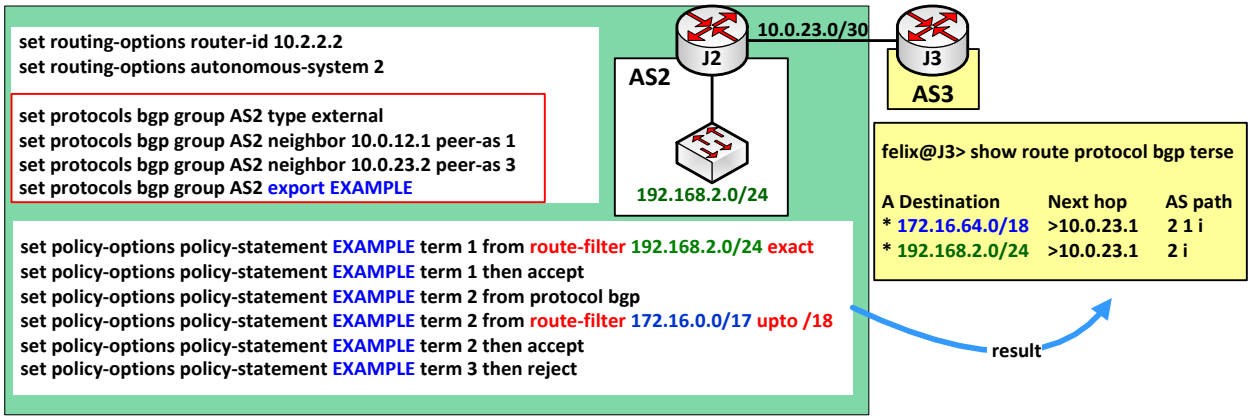


Figure 5.41

Explanation:

- The policy matches 192.168.2.0/24 exactly coming from directly connected.
- The policy also matches the routes advertised by BGP that have the prefix 172.16.0 all the way up to the bit 17 and then any other combination happening until bit 18.
- Two possible prefixes would comply with these requirements 172.16.0.0/18 and 172.16.64.0/18, but 172.16.0.0/18 is not even present in this configuration.
- The prefix available is 172.16.64.0/18. The next figure illustrates the point:

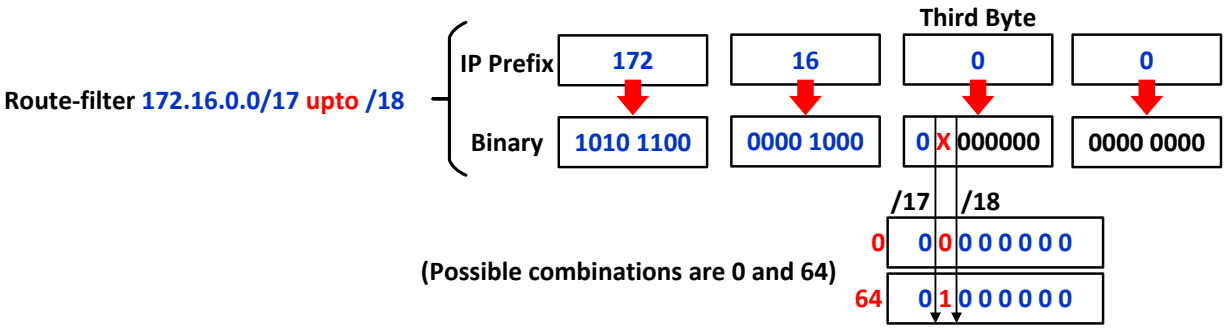


Figure 5.40

Statement: in J2 accept the prefixes that begin with 172.16. that have masks between /21 and /24.

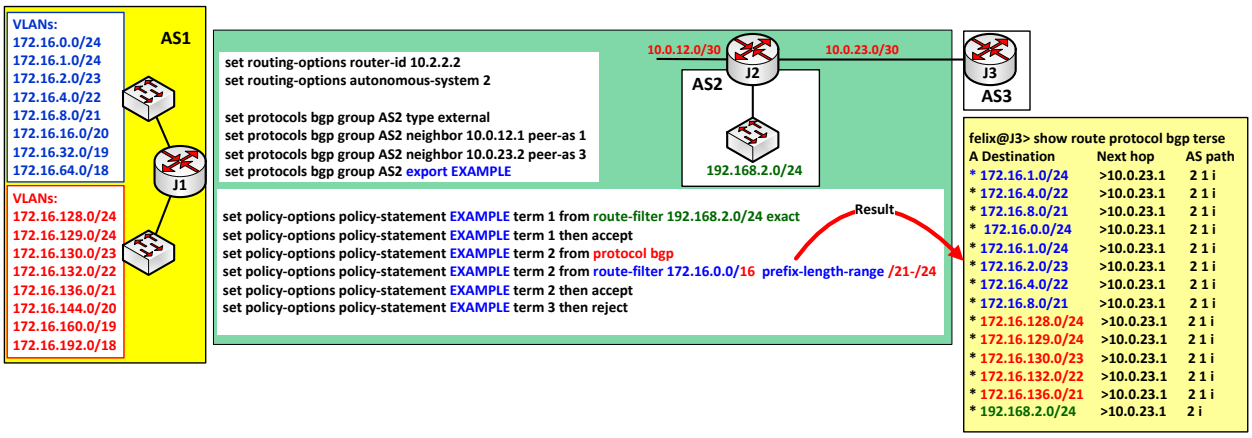


Figure 5.41

Statement: in J2 accept the prefixes that begin with 172.16.128 that have masks between /17 and /22.

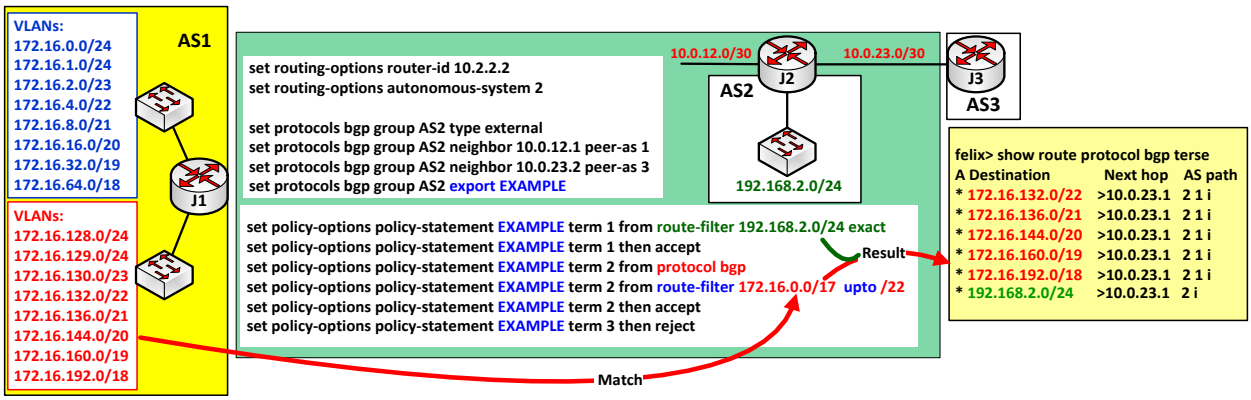


Figure 5.42

Routing Policy Specification Language RPSL

Internet domains have different BGP peering profiles depending of the nature and position of the organizations involved. There are many peering models, for example:

- For an enterprise, the basic need is to just get access to the Internet by peering with one or more Internet Service Providers ISP. An enterprise typical routing policy consist of handling MED, LOCAL_PREF, and AS_PATH attributes.
- In turn, the ISPs need to peer with other well connected Internet organizations that act as transit providers to the rest of the Internet. Typically, an ISP implement policies based on MED, LOCAL_PREF and a lot of COMMUNITIES.
- The Internet carriers peer with each other and with the ISPs and in most cases they also have their own customers. They also deal with COMMUNITIES and AGGREGATION of traffic.

Regardless of the position and structure of the organizations, they all have one common and critical necessity which is to state their **routing intentions** in a clear and concise way and to understand the routing intentions of the neighbours.

For instance, a massive Internet company such as Cogent connects with 5940 AS (as per July 2017 [3]). It would be obviously unmanageable that Cogent were to negotiate its routing intentions, from scratch, with every one of these AS. Instead, the routing policies must be stated in an industry format that is clearly understood and openly accessible.

Routing Policy Specification Language is a formal language designed [4, 5] to fulfill the need of Internet domains to describe their routing policies in a universal, standard and open source way. The policies are written in RPSL and stored in public databases maintained by the Internet Routing Registries [5] such as RIPE, RADB, APNIC, CANARIE and others.

These databases are implemented in whois format (whois is a query/response style protocol used to access databases). Therefore, AS administrators can retrieve policy information from these registries. From the obtained data, router configurations can be derived from the reading and interpretation of an RPSL statements. However, RPSL does not provide a method to produce the configurations. That still has to be done with some configuration [6] tool. In short, RPSL is not a configuration program; however, routing configuration can be derived from the specifications of an RPSL policy which is fundamentally pseudo-code.

RPSL structure

Languages have **syntax** and **semantics**. Syntax is the set of rules for organizing the words to create well-formed sentences in a language. Semantics is the meaning of the words and sentences. Hence, RPSL provides the syntax and the semantics to describe the routing requirements of an Autonomous System.

RPSL is an Object Oriented language, which means that every possible item that could appear in a routing policy is described as an Object with Attributes. Depending of the nature of the objects, they are grouped in Classes which act as an organizational hierarchy. These classes are defined by the logical actions typically needed in a routing domain, for example:

Concept	Class
Organization that can write, delete or modify	**mntner** (maintainer)
People who are the administrators	**person**
People who are the administrators	**role**
Autonomous Systems information	**aut-num**
IP Routes	**route**
Objects related to AS	**as-set**
Items subject to filtering, typically routes.	**route-set**
Items subject to filtering, typically routes.	**filter-set**
Applied to a peer	**peering-set**
Description of a router	**inet-rtr**
	rtr-set
	dictionary

The syntax of the language has rules such as:

- Only ASCII characters can be used.
- Object names must start with a letter.
- Object names are not case sensitive.
- The words that are needed in the routing policies are reserved

Also, there are reserved words for special meaning:

accept	any	except	networks	peeras
action	as-any	from	not	refine
and	at	inbound	or	rs-any
announce	atomic	into	outbound	to

Like any language, RPSL has an extensive structure and vocabulary thoroughly described in the Request for Comments RFC-2622 and RFC-2650. Nevertheless, the previous fundamentals are barely enough to start explaining by examples.

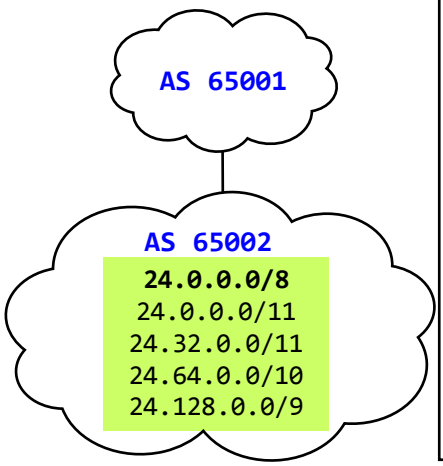
Describing Policies with RPSL

Disclaimer: the information used in the examples is of public access in the Internet and it is being used only for educational uses. The intention is to try to portray real life situations but they are not necessarily current or accurate.

RPSL Examples

Simple peering, advertising addresses

AS 65001 peers with AS65001 to advertise its block of addresses summarized by 24.0.0.0/8.




```
aut-num:   AS65002
as-name:   Example-net
descr:     An example network
admin-c:   admin-example
mnt-by:    Example
route:     24.0.0.0/8
origin:    AS65002
import:    from AS65001 accept ANY
export:    to AS65001 announce { 24.0.0.0/8 }
```

Figure 5.43

The previous RPSL statements only have the most critical fields for a basic peering. The route attribute specifies the address space and the origin attribute specifies the autonomous system. AS65002 is accepting any routing information from AS65001. Regarding the export statement, there can be several variations depending of the intentions of the administrators, for example:

- Advertise only the subnets of the block without the aggregation 24.0.0.0/8

```
export: to AS65001 announce { 24.0.0.0/8^- }
```

The regular expression **^-** means exclude the prefix 24.0.0.0/8 but allow every subnet of it.

- Advertise only the subnets of the block including the aggregation 24.0.0.0/8

```
export: to AS65001 announce { 24.0.0.0/8^+ }
```

The regular expression **^+** means include the prefix 24.0.0.0/8 and its subnets. So the 5 addresses would be advertised.

- Advertise only the subnets with a specific mask.

```
export: to AS65001 announce { 24.0.0.0/8^11 }
```

The regular expression **^11** matches the subnets of prefix 24.0.0.0/8 with a mask length of /11. That will take 24.0.0.0/11 and 24.32.0.0/11 only.

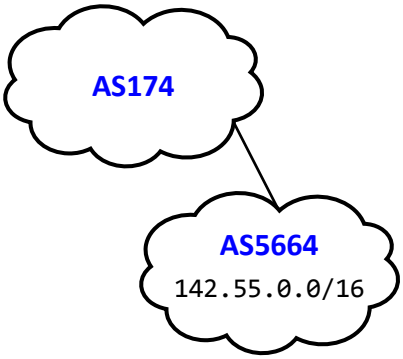
- Advertise the subnets with a masks within a range.

```
export: to AS65001 announce { 24.0.0.0/8^9-10 }
```

The regular expression **^9-10** matches the subnets of prefix 24.0.0.0/8 with a mask length between 9 and 10 inclusive. That will take 24.64.0.0/9 and 24.192.0.0/10 only.

Client organization accessing Internet

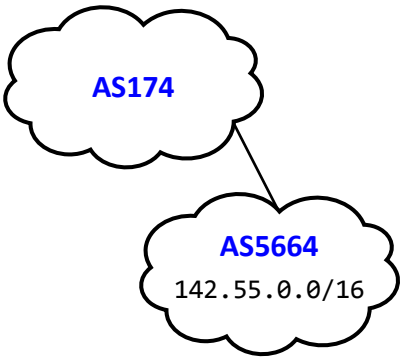
An organization gains access to the Internet via a large carrier, in this case Sheridan College AS5664 via Cogent AS174. In its most simple format the RPSL description is shown with some of the most important attributes:




```
aut-num:   AS5664
as-name:   Sheridan College
descr:     Sheridan College, Ontario
admin-c:   admin-Shernet
mnt-by:    Shernet
route:     142.55.0.0/16
origin:    AS5664
import:    from AS174 accept ANY
export:    to AS174 announce <^AS564$>
```

Figure 5.44

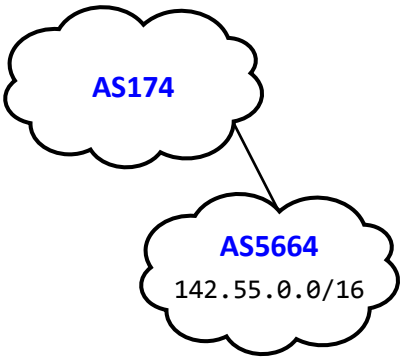
That is a generic description where the customer advertises its IPv4 prefix. Sheridan also describes accepting any routing prefix from the provider; basically any routing information that Cogent would send to Sheridan. Suppose that the provider wants to make sure that only the prefix range owned by the customer is accepted.




```
aut-num:   AS174
as-name:   Cogent
descr:     Cogent/PSI
import:    from AS5664 accept { 142.55.0.0/16 }
export:    to AS5664 announce ANY
```

Figure 5.45

The same routing intention can be expressed with a match with the regular expression ^AS564$ to the routes sourced in the neighbour autonomous system (however this declaration would accept any prefix sourced by AS5664).

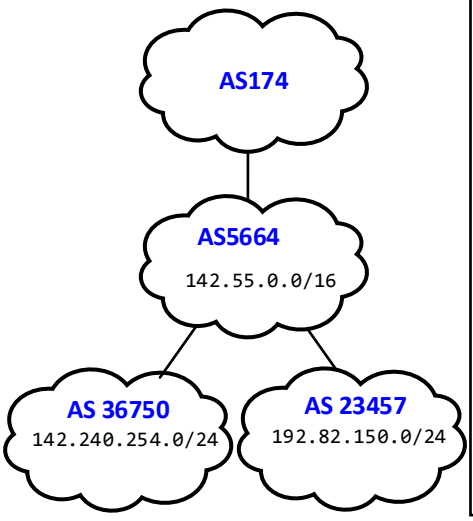



```
aut-num:   AS174
as-name:   Cogent
descr:     Cogent/PSI
import:    from AS5664 accept <^AS5664$>
export:    to AS5664 announce ANY
```

Figure 5.46

Let's add new elements to the previous topology. Sheridan provides transit to parts of the networks of the City of Mississauga (AS36750) and the City of Brampton (AS23457) as per collaboration agreements. In this case, the RPSL declaration must be modified.

AS5664 could accept the prefix 142.240.254.0/24 from the City of Mississauga and 192.82.150.0/24 from the City of Brampton. Then AS5664 would advertise the AS_PATH to reach both prefixes to AS174 (Cogent). This would be done with regular expressions characters to indicate the beginning and end of the strings.




```
aut-num: AS5664
as-name: Sheridan College
descr:   Sheridan College, Ontario
admin-c: admin-Shernet
mnt-by:  Shernet
route:   142.55.0.0/16
origin:  AS5664
import:  from AS174 accept ANY
export:  to AS174 announce { 142.55.0.0/16}
export:  to AS174 announce <^AS5664 AS36750$>
export:  to AS174 announce <^AS5664 AS23457$>
import:  from AS36750 accept { 142.240.254.0/24 }
export:  to AS36750 announce ANY
import:  from AS23457 accept { 192.82.150.0/24 }
export:  to AS23457 announce ANY
```

Figure 5.47

Internet Provider to client

On the side of the carrier, this is the actual peering declaration of Cogent with respect Sheridan College according to a search executed [7] in the RIPE database on July 2017.

```
aut-num: AS174
import: from AS5664 accept ANY
export: to AS5664 announce ANY
mp-export: afi ipv6.unicast to AS5664 announce ANY
mp-import: afi ipv6.unicast from AS5664 accept ANY
```

Figure 5.48

Cogent has thousands of peers, so it rather simplifies its policy declaration to accepting and advertising any IPv4 prefixes. Cogent also declares that it supports multiprotocol (MP) with the address family (AFI) IPv6 routes even though the client, in this case, does not have any IPv6 internally. If the customer eventually implements IPv6, then the policy is ready to accept that.

As an interesting fact, the routing policy appears in the RIPE database in three different formats: RPSL, JSON (Java Script Object Notation) and HTML. Some organizations have in-house programing tools to process the data and they choose one of these formats according to their preferences. This is Cogent's routing policy with respect Sheridan College in JSON format:

```
{
      "name" : "import",
      "value" : "from AS5564 accept ANY"
    }, {
      "name" : "export",
      "value" : "to AS5564 announce ANY"
    }, {
      "name" : "mp-import",
      "value" : "afi ipv6.unicast from AS5564 accept ANY"
    }, {
      "name" : "mp-export",
      "value" : "afi ipv6.unicast to AS5564 announce ANY"
    }
```

This is the same policy in Hypertext Markup Language HTML:

```
<attribute name="import" value="from AS5664 accept ANY"/>
<attribute name="export" value="to AS5664 announce ANY"/>
<attribute name="mp-import" value="afi ipv6.unicast from AS5664 accept ANY"/>
<attribute name="mp-export" value="afi ipv6.unicast to AS5664 announce ANY"/>
```

Transit for Collaboration Network

The following example is inspired in the organization of the Canadian National Research and Education Network [8] which is a publicly funded conglomerate of universities, colleges, research centres, health organizations and government. This network relies upon a national optical fibre backbone operated by Canarie (AS6509) [8] with interconnected regional networks such as the Ontario Research and Innovation Network (Orion AS26677) shown in the schematics below.

The main point of this network is to allow collaboration among its members and to access the resources located in domains such as Orion and Canarie (data centres for computing research data, for example). The five members shown in the diagram do not access the rest of Internet via Orion but rather they all have commercial providers (not shown) for Internet access.

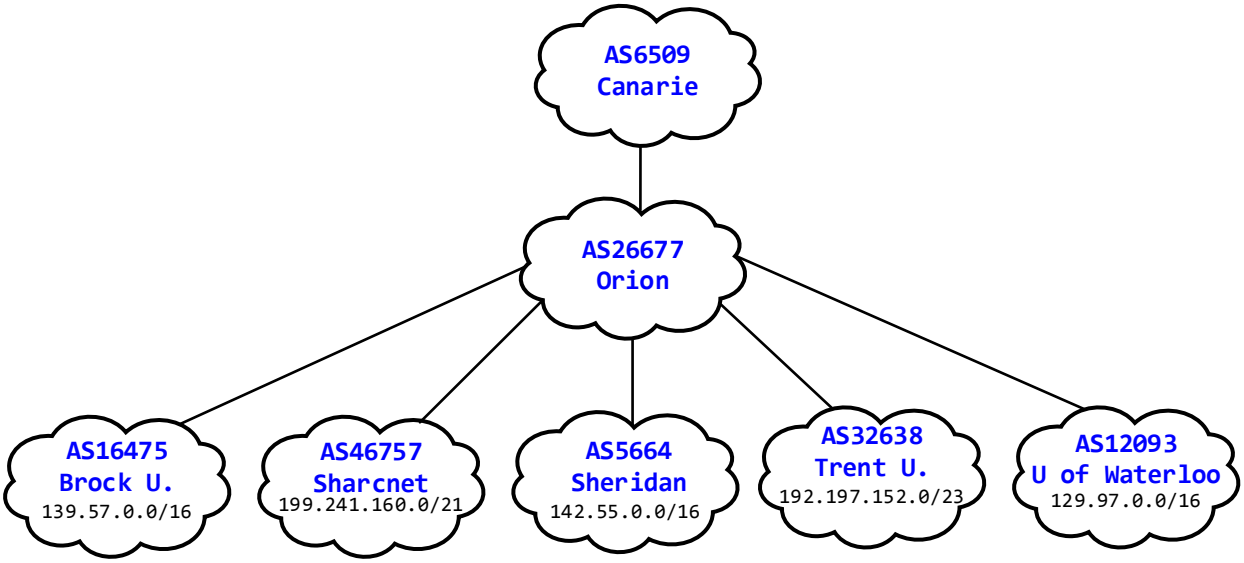


Figure 5.48

Based on this declaration of routing principles:

- Orion should advertise its own network prefixes to the members.
- Orion should advertise Canarie's network prefixes to the members.
- Orion should not advertise access to the rest of the Internet to its members.
- Orion should advertise the individual member's networks to each other.

Reminder note: this is a made up example, Orion does not necessarily do this in real life.

Implementing the previous policy intent can become cumbersome as Orion a list of several dozens of members that keeps growing with more organizations being added. But, since all the members have common factors, the RPSL description can be simplified using the syntactic object as-set. The format of the as-set [5] describes the parent or transit AS (Orion AS26677 in this case) and a general AS-CUSTOMERS field [5] which is implicitly described by the members object.

```
aut-num: AS26677
as-name: Orion
descr:   Ontario Research Innovation Optical Network
as-set:  AS26677:AS-CUSTOMERS
members: AS16475 AS46757 AS5664 AS32638 AS12093
export: to AS6509 announce AS26677:AS-CUSTOMERS
export: to AS26677:AS-CUSTOMERS announce AS26677 AS6509
import: from AS26677:AS-CUSTOMERS accept <^AS26677:AS-CUSTOMERS$>
```

Figure 5.49

The advantage of this system is that new clients just need to be added to the list of members and the export and import statements are simplified also. For instance, instead of listing each AS number in the import statement, the string `^AS26677:AS-CUSTOMERS$` is used. The regular expression characters ^ and $ delimit that only the AS listed in the members list are accepted. On the other hand, all the customers receives the information about the AS26677 and AS6509.

Chapter Summary

The policy engine is what gives BGP a distinctive edge. Traffic interests and commercial pacts can be stated in a document and then translated into a configuration file so that BGP can implement these routing intentions. The policy engine is a core BGP function and it is critical for the network engineer to understand how it works and how to use it.

This chapter, started with the default BGP algorithm that chooses the best path among several routes. Then the routing engine structure was explained as a system of databases to store routing information and the policy engine that makes decisions based on policy rules. After that, routing policies were implemented in both Cisco and Juniper operating systems to filter and to modify basic attributes that control traffic flows.

Routing Policy Specification Language RPSL is an initiative to provide the users of BGP with a language to express routing intentions. Complex policies can be written in RPSL documents that get posted in Internet Routing Registries where they are available via whois queries.

Exercise

The following topology is meant to test the concepts of route filtering in both Cisco and Juniper.

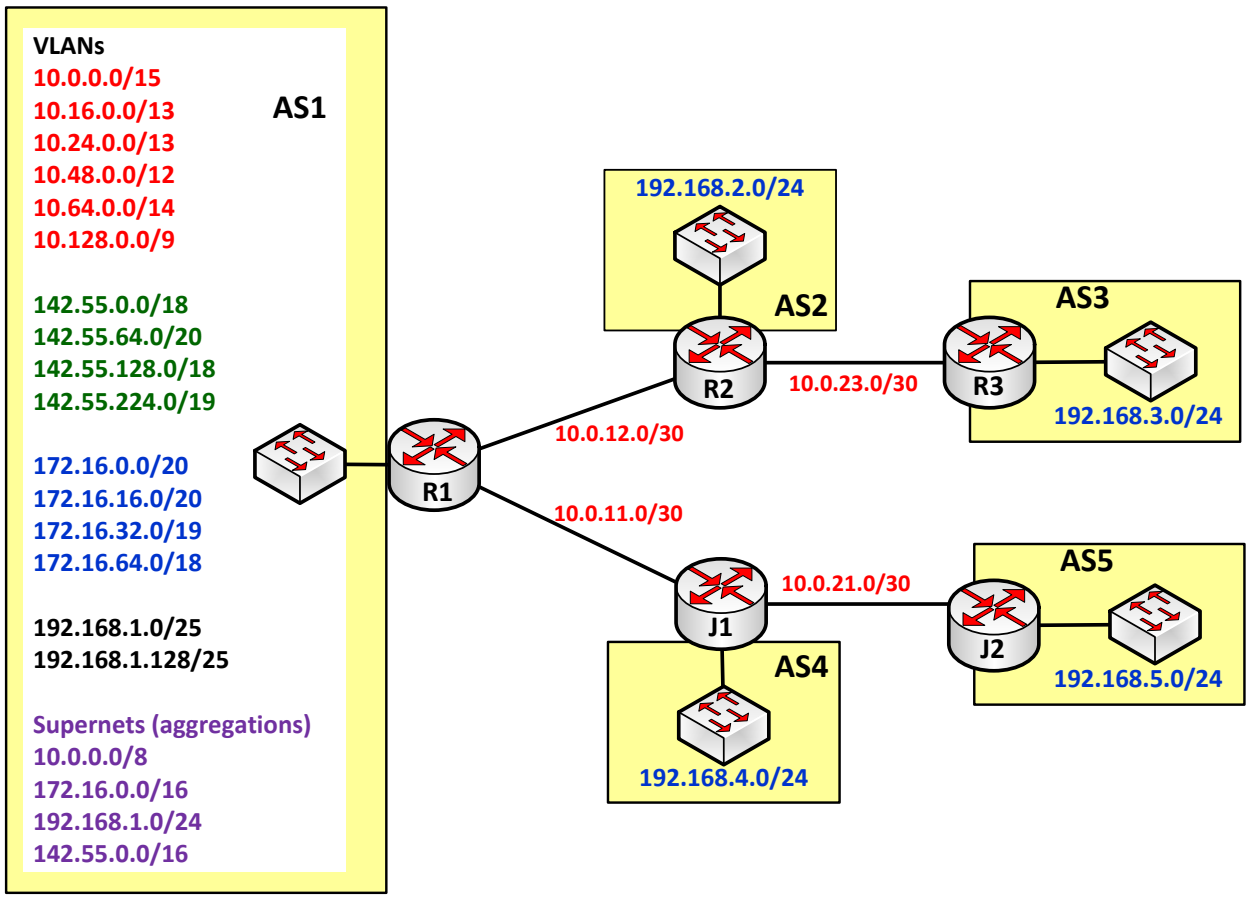


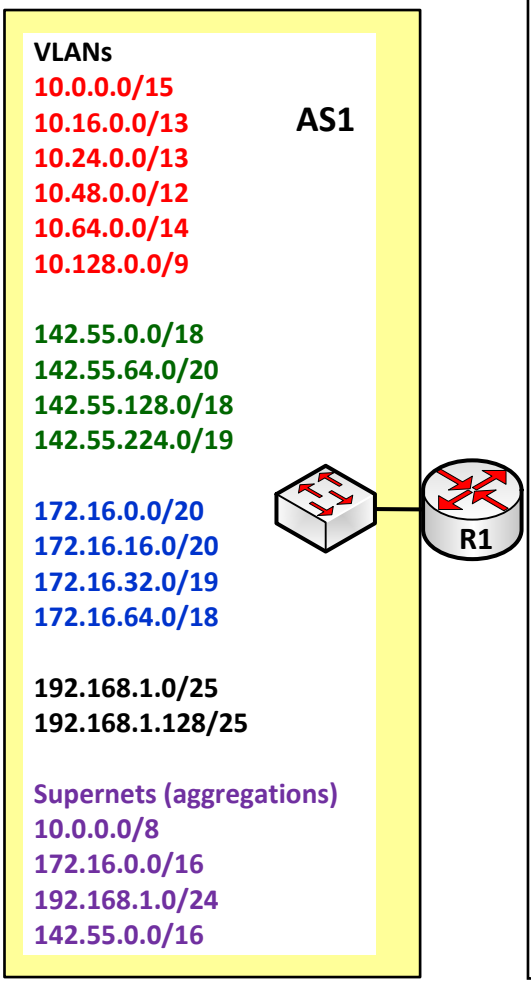
Figure 5.50

Topology

- Router R1 "feeds" routing information via BGP to AS2 and AS4.
- This routing information has been designed to contain prefixes with a variety of masks.
- Router R1 acts as a "routing table feeder" basically. It is just injecting interesting routing information to its neighbours AS.
- Sixteen (16) VLANs has been configured toward SW1 and their configuration is posted below.
- Four (4) aggregations have been done in R1.
- R1's configuration VLANs file is provided next:

Router R1 VLANs

```
interface f0/0
no shutdown
!
interface f0/0.1
 encapsulation dot1q 1 native
 ip address 10.0.0.1 255.254.0.0
!
interface f0/0.2
 encapsulation dot1q 2
 ip address 10.16.0.1 255.248.0.0
!
interface f0/0.3
 encapsulation dot1q 3
 ip address 10.24.0.1 255.248.0.0
!
interface f0/0.4
 encapsulation dot1q 4
 ip address 10.48.0.1 255.240.0.0
!
interface f0/0.5
 encapsulation dot1q 5
 ip address 10.64.0.1 255.252.0.0
!
interface f0/0.6
 encapsulation dot1q 6
 ip address 10.128.0.1 255.128.0.0
!
interface f0/0.7
 encapsulation dot1q 7
 ip address 172.16.0.1 255.255.240.0
!
interface f0/0.8
 encapsulation dot1q 8
 ip address 172.16.32.1 255.255.224.0
!
interface f0/0.9
 encapsulation dot1q 9
 ip address 172.16.16.1 255.255.240.0
!
interface f0/0.10
 encapsulation dot1q 10
 ip address 172.16.64.1 255.255.192.0
!
interface f0/0.11
 encapsulation dot1q 11
 ip address 192.168.1.1 255.255.255.128
!
interface f0/0.12
 encapsulation dot1q 12
 ip address 192.168.1.129 255.255.255.128
!
interface f0/0.19
 encapsulation dot1q 19
 ip address 192.168.2.0 255.255.255.0
!
interface f0/0.20
 encapsulation dot1q 20
 ip address 192.168.3.0 255.255.255.0
!
interface f0/0.15
 encapsulation dot1q 15
 ip address 142.55.0.1 255.255.192.0
!
interface f0/0.16
 encapsulation dot1q 16
 ip address 142.55.64.1 255.255.240.0
!
interface f0/0.17
 encapsulation dot1q 17
 ip address 142.55.128.1 255.255.192.0
!
interface f0/0.18
 encapsulation dot1q 18
 ip address 142.55.224.1 255.255.224.0
```



R1 BGP Commands

```
router bgp 1
 network 10.0.0.0 mask 255.254.0.0
 network 10.16.0.0 mask 255.248.0.0
 network 10.24.0.0 mask 255.248.0.0
 network 10.48.0.0 mask 255.240.0.0
 network 10.64.0.0 mask 255.252.0.0
 network 10.128.0.0 mask 255.128.0.0
 network 142.55.0.0 mask 255.255.192.0
 network 142.55.64.0 mask 255.255.240.0
 network 142.55.128.0 mask 255.255.192.0
 network 142.55.224.0 mask 255.255.224.0
 network 172.16.0.0 mask 255.255.240.0
 network 172.16.16.0 mask 255.255.240.0
 network 172.16.32.0 mask 255.255.224.0
 network 172.16.64.0 mask 255.255.192.0
 network 192.168.1.0 mask 255.255.255.128
 network 192.168.1.128 mask 255.255.255.128
 aggregate-address 10.0.0.0 255.0.0.0
 aggregate-address 192.168.1.0 255.255.255.0
 aggregate-address 172.16.0.0 255.255.0.0
 aggregate-address 142.55.0.0 255.255.0.0
 neighbor 10.0.11.2 remote-as 4
 neighbor 10.0.12.2 remote-as 2
 no auto-summary
```

Figure 5.51

Policies

- Construct and apply prefix-lists or route-filter to implement the following policies and test them each separately in the previous topology.
- Design each policy in R2 to be applied outbound toward AS3. Observe results in R3.
- Design the same policy in J1 to be applied outbound toward AS5. Observe results in J2.
- Permit only 192.168.1.0/24
- Permit only 192.168.1.0/24 and 192.168.1.0/25
- Permit only addresses that begin with 192 exactly.
- Deny all the 192.168.0.0 and then it allows everything else.
- Deny the routes that begin with 10, but it allows everything else.
- Permit 172.16.0.0 exactly and any of its "subnets". Deny everything else.
- Permit the first byte of 10.0.0.0 with any mask between /16 and /24. Deny else.
- Deny routes with the first byte of 10.0.0.0 but with any mask with a length between /16 and /24. Allow everything else.
- Deny all private IP address space. Allow else.
- Permit only (formerly called) class-full IP address space. Deny else.
- Permit only (so called) "subnets". Deny else.
- Permit prefixes 142.55.0.0 with masks between /17 and /18. Deny else.
- Permit prefixes 142.55.0.0 with masks within /17 and /19. Deny else.
- Permit prefixes 142.55.0.0 with masks upto /20.

Implement the following network topology. The routers labeled R should be Cisco and the routers labeled J should be Juniper. Implement the policies outlined below.

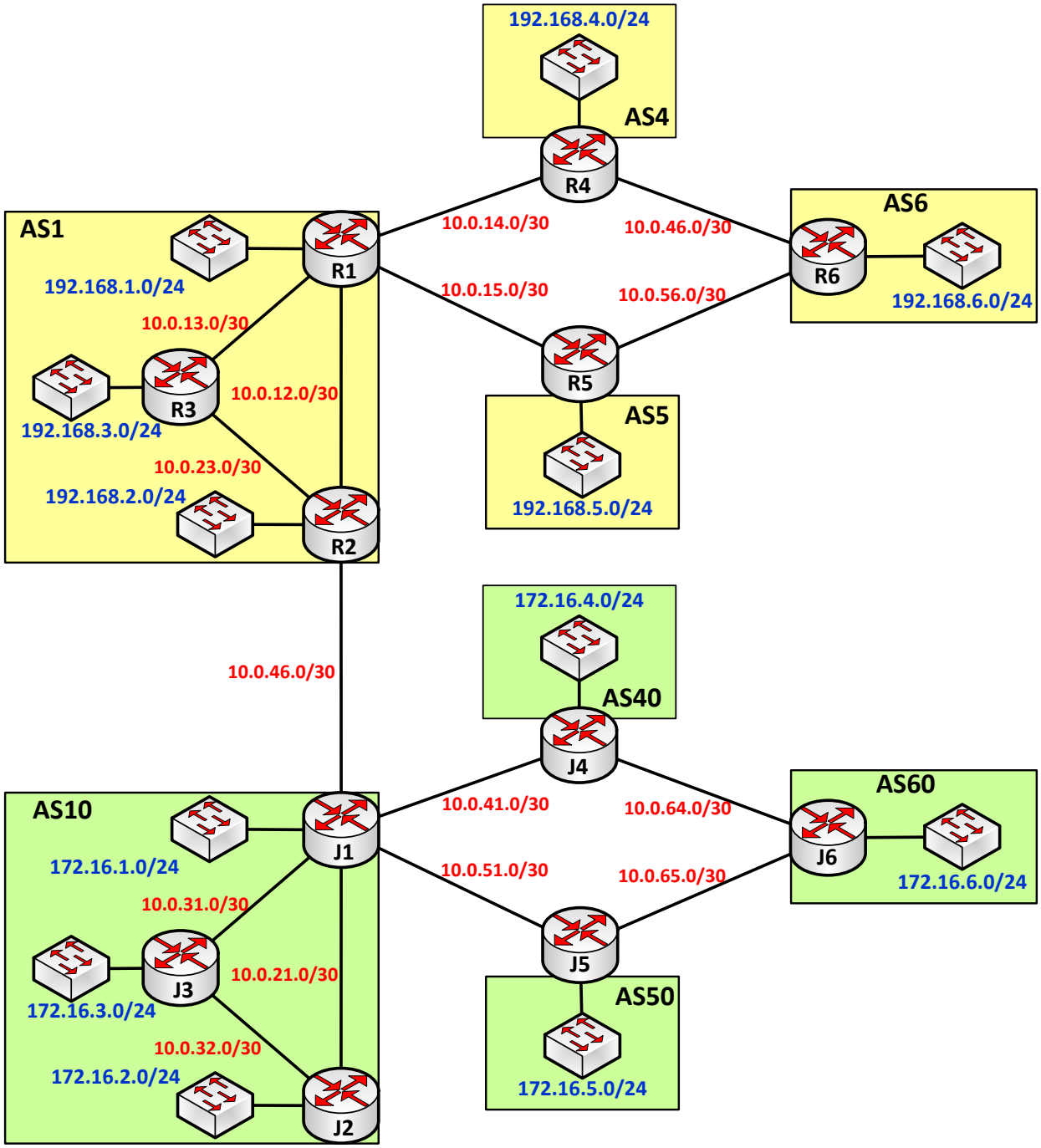


Figure 5.60

In router R1 implement the following policies and apply them outbound to R4.

- Permit routes sourced from AS1. Deny else.
- Permit routes sourced from either AS1 or AS10. Deny else.
- Permit routes sourced from either AS1 or AS10 or AS50. Deny else.
- Permit routes that have AS40 in the path. Deny else.
- Deny routes sourced from AS10, but allow anything that has AS10 in the path.
- Permit routes sourced from either AS5 or AS40. Deny else.
- Permit routes that have either AS5 or AS50 in the path. Deny else.
- Permit routes that have either AS5 or AS10 or AS40 in the path. Deny else.
- Permit routes that have the AS_PATH 10 50 60 only.

In router J1 implement the following policies and apply them outbound to J4.

- Permit routes sourced from AS1. Deny else.
- Permit routes sourced from either AS1 or AS10. Deny else.
- Permit routes sourced from either AS1 or AS6 or AS10 or AS50. Deny else.
- Permit routes that have AS4 in the path. Deny else.
- Permit routes sourced from either AS4 or AS5. Deny else.
- Deny routes sourced from AS1, but allow anything that has AS1 in the path.
- Permit routes that have either AS5 or AS50 in the path. Deny else.
- Permit routes that have either AS1 or AS4 or AS50 in the path. Deny else.
- Permit routes that have the AS_PATH 10 50 60 only.

Implement the following network topology. The routers labeled R should be Cisco and the routers labeled J should be Juniper.

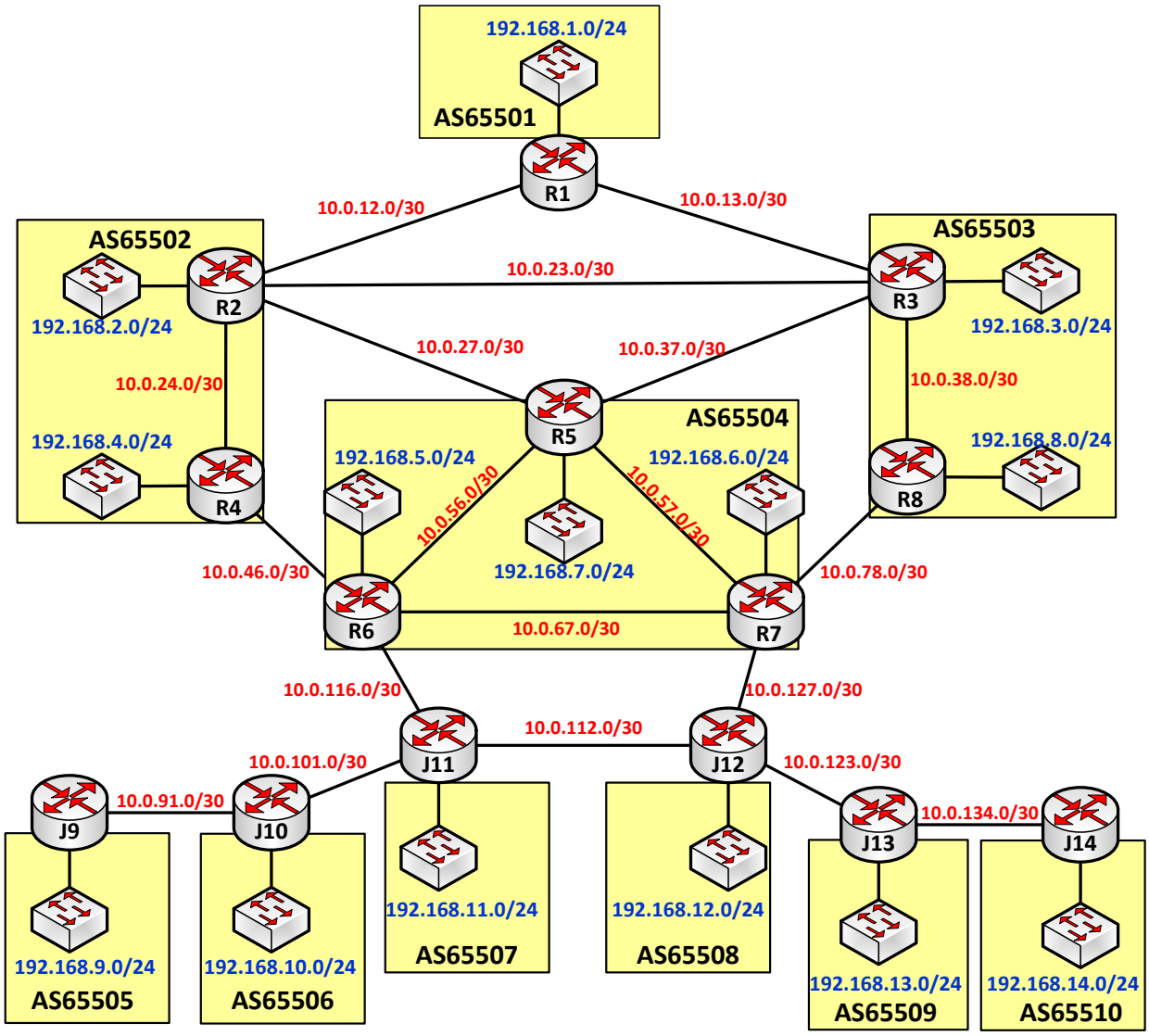


Figure 5.61

In the Cisco router R1, issue show commands with the following regular expressions. Compare and analyze the results.

- show ip route regexp **6550.$**
- show ip route regexp **655.0$**
- show ip route regexp **655..$**
- show ip route regexp **655.+$**
- show ip route regexp **655[1-3]$**
- show ip route regexp **_65501_|_65503_**
- show ip route regexp **6$**
- show ip route regexp **50.$**
- show ip route regexp **55.$**
- show ip route regexp **55.+$**
- show ip route regexp **55.+**
- show ip route regexp **4.+**
- show ip route regexp **4.+$**
- show ip route regexp **^4.+**
- show ip route regexp **^3.+**
- show ip route regexp **^6550.**
- show ip route regexp **^6550.***
- show ip route regexp **^6550.+**
- show ip route regexp **^655.0**
- show ip route regexp **^6...3_6..1.$**
- show ip route regexp **^6...3_.*_6510.$**
- show ip route regexp **^6...3_.*_6..1.$**
- show ip route regexp **^6...3_._6..1.$**
- show ip route regexp **^6...3_.+_6..1.$**
- show ip route regexp **^65503_65504_.*_65510$**
- show ip route regexp **^65503_6550[1-5]_.*_6..1.$**
- show ip route regexp **^65503_[0-95...]*$**

In Juniper J9, issue show commands with the following regular expressions. Compare and analyze the results.

- show route terse aspath-regex **^65504$**
- show route terse aspath-regex **65504**
- show route terse aspath-regex **.65504.**
- show route terse aspath-regex **"65502 65504"**
- show route terse aspath-regex **"65503 65504"**
- show route terse aspath-regex **".+65502 65504.+"**
- show route terse aspath-regex **"65503. 65504"**

RPSL

This is the Laboratory S144 used to teach BGP at Sheridan College's Internet Communications. The top AS 65144 advertises a full Internet routing table to AS65145 which in turn, re-advertises the table to the autonomous systems at the lower level. Each AS is owned by a student, for example AS65010 belongs to the first student in the class list, AS65020 to the second and so on, so forth.

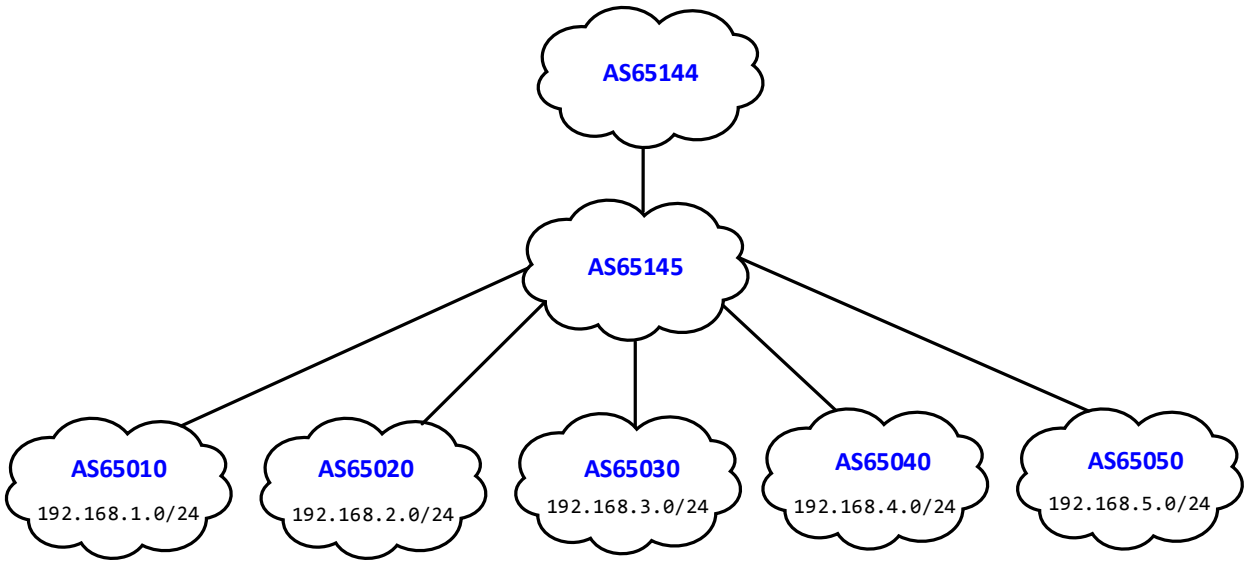


Figure 5.62

- Write the RPSL documents for AS65144, AS65145 and one of the students owned AS.
- In the RPSL document for AS65145, group all the students' AS in one as-set of customers.

This is an expansion of the previous scenario. Each student has 10 autonomous systems to practice routing policies. The first student in the class list has AS numbers 65010, 65011, 65012, 65013... all the way to 65019. The second student gets the AS numbers 65020 all the way to 65029. The same pattern is repeated for each student in the class. Only the first AS of each student peers with AS65145. The other AS have different arrangements but they always appear, in some order, after the first AS.

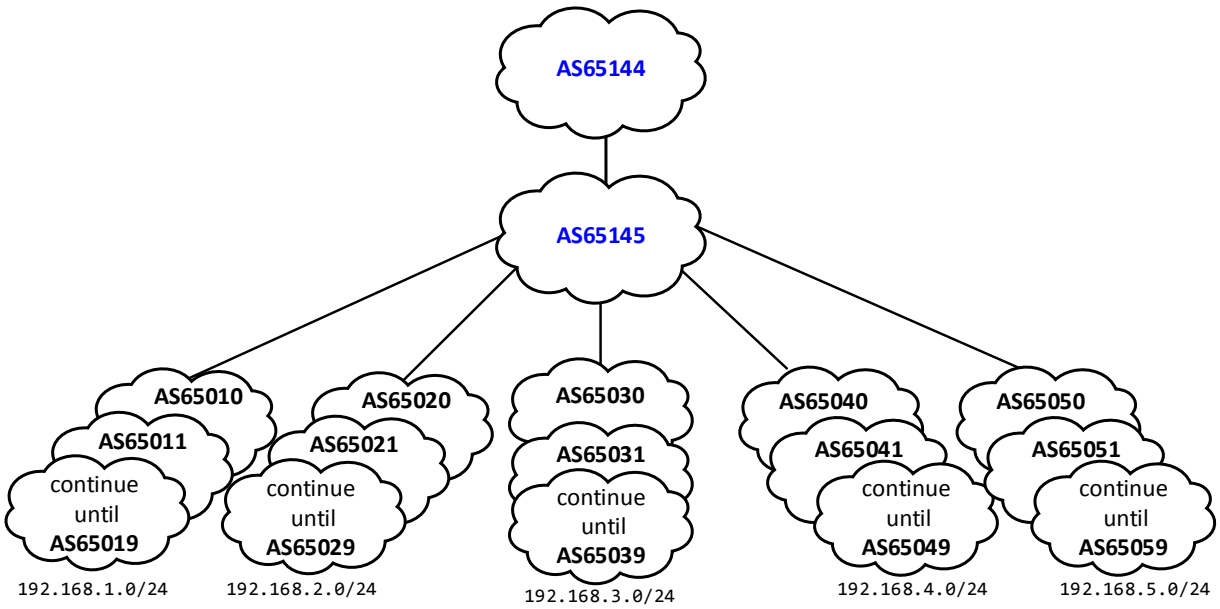


Figure 5.63

- Write an RPSL document for AS65145 that will consider each student as an as-set.
- Then group all these as-set in another as-set called S144
- Use regular expressions to make the description more efficient trying to match as many common patterns as possible.

Chapter References

[1] www.juniper.net/techpubs/software/junos/junos74/swconfig74-policy

[2] Randy Zhang and Micah Bartell. BGP Design and Implementation. Ciscopress.com.

[3] http://www.cogentco.com/en/network/connected-autonomous-systems. (July 017)

[4] C. Alaettinoglu, C. Villamizar, E. Gerich, D. Kessens, D. Meyer, T. Bates, D. Karrenberg, and M Terpstra. Request for Comments RFC-2622 Routing Policy Specification Language RPSL (1999).

[5] D. Meyer, J. Schmitz, C. Orange, M. Prior. C. Alaettinoglu. Request for Comments RFC-2650 Using RPSL in Practice. (1999).

[5] Internet Routing Registries. http://www.irr.net/docs/list.html. (July 2017)

[6] IRRtoolset, RtConfig tool. https://github.com/irrtoolset/irrtoolset/wiki. (July 2017)

[7] RIPE Database Search. https://apps.db.ripe.net/search/ (July 2017)

[8] Canarie. https://www.canarie.ca/about-us/ (July 2017)

CHAPTER 6

Control of Traffic between AS

The most common BGP policy is the control of inbound and outbound traffic flows between autonomous systems. BGP has two attributes MED and LOCAL-PREF that are typically manipulated to implement such traffic control policies. Additionally, the AS-PATH can be made more or less appealing with the technique of prepending AS numbers. This section covers these aspects of the BGP protocol.

Default Behaviour of a Network Topology

The network topology below shows two routing domains peering with each other. Before getting into the configuration of policies, it is convenient to analyze this network to understand its functioning. Autonomous system 1 contains Cisco only routers and they run EIGRP to build the IP connectivity inside the domain. Additionally, iBGP is configured to provide router R1 with visibility about the external IP routes that will be re-advertised by the AS border routers (ASBR) R2 and R3. In turn, these two ASBRs learn the external routes from the neighbour AS number 2. Also, these two border routers advertise outbound the internal routes 192.168.1.0/24, 192.168.2.0/24 and 192.168.3.0/24 with the use of the (Cisco) network command under the BGP configuration.

Autonomous system 2 contains Juniper only routers and they run OSPF inside this AS. The ASBRs Juniper J4 and J5 advertise the internal prefixes 192.168.4.0/24, 192.168.5.0/24 and 192.168.6.0/24. Routers J4, J5, and J6 also run iBGP to have a complete vision of the external routes within AS2. The following diagram shows the network topology. The complete basic, no policies, configuration files are included in the appendix at the end of this chapter.

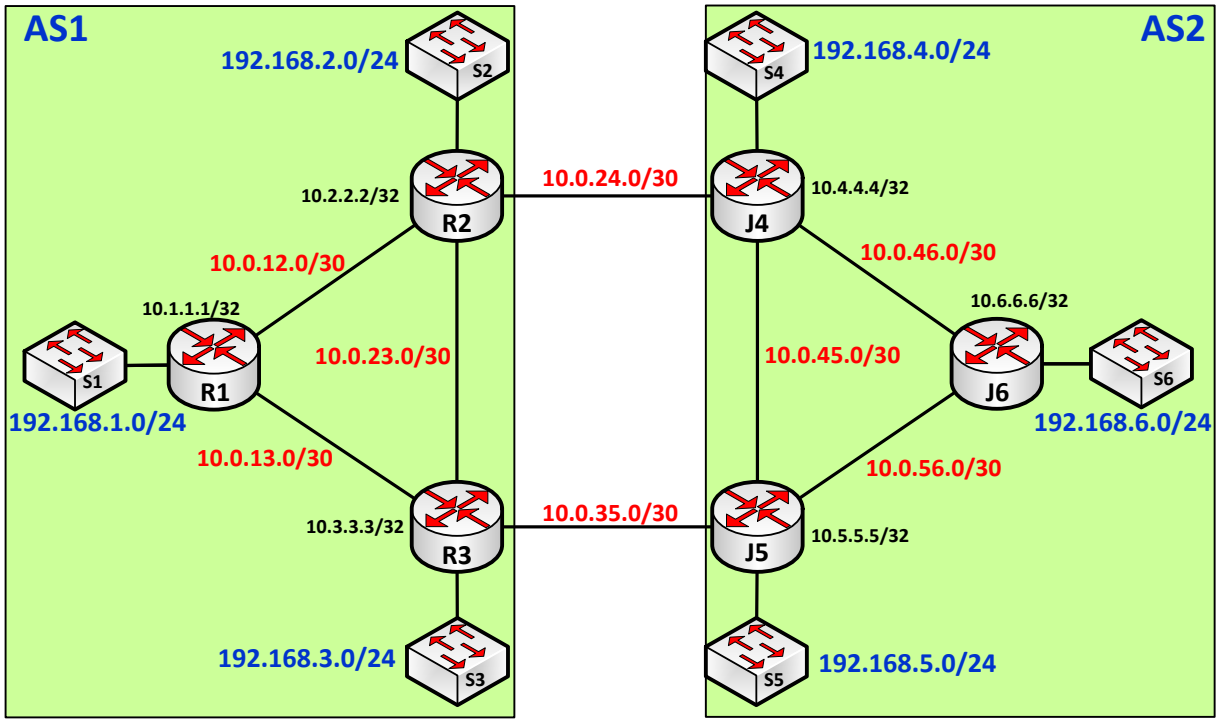


Figure 6.1

This topology has been purposely designed to showcase the properties of MED and LOCAL-PREF. To that effect, it is important to notice in the figure that the physical paths between the networks 192.168.1.0/24 and 192.168.6.0/24 are symmetrical. Let's observe this effect in router R1. The following diagram represents the information stored inside the IP routing table and the BGP database after the network has converged.

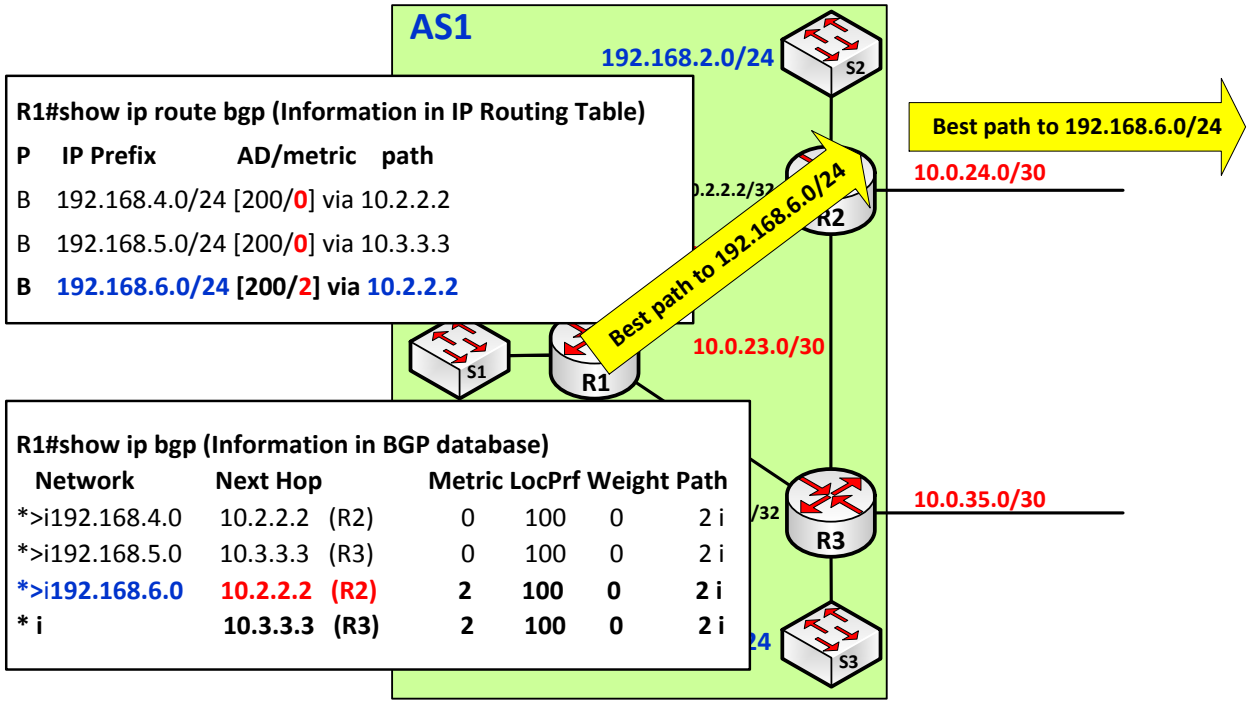


Figure 6.2

In the routing table, the **Administrative Distance** of **200** for all the routes indicates that were announced by iBGP. That makes sense, since R1 only peers internally with R2 and R3. Also, all the routes appear with a **LOCAL PREFERENCE** of 100 which is the default value for routes learned via iBGP.

In Cisco IOS, the **metric** in the BGP database has the same value as the **MED**. It will be observed later than JUNOS has a different approach to this. Also, in Cisco IOS, there is a proprietary value called the **Weight**. In the case of R1, all the routes have a weight of zero meaning that there is no preference in particular. (Cisco Weight is not a BGP attribute).

The BGP database shows that there are two valid paths to reach prefix 192.168.6.0/24 from R1's standpoint. One path goes from R1 toward R2, and the other path goes from R1 toward R3. Because both routes are equally good, a tie breaker is needed in the decision process. Thus, the BGP algorithm chooses, after going thru several steps, the lowest router-id between 10.2.2.2 and 10.3.3.3.

The practical result is that all the traffic leaving R1 with destination 192.168.6.0/24 will go to router R2 via 10.24.24.2 by default. Traffic destined to 192.168.4.0/24 will also use the same path. In fact, the link between routers R3 and J5 is only used to reach the network

192.168.5.0/24. Consequently, the other link might be more saturated with traffic. Typically, this kind of situation requires traffic manipulation and that makes this topology a good case to explain these policies.

Further examination of routing information in the border router R2 verifies that this router prefers to reach 192.168.4.0/24 and 192.168.6.0/24 via the direct peer AS2-JUNOS4 while the path to reach 192.168.5.0/24 goes to the iBGP neighbour R3. Let's find out why.

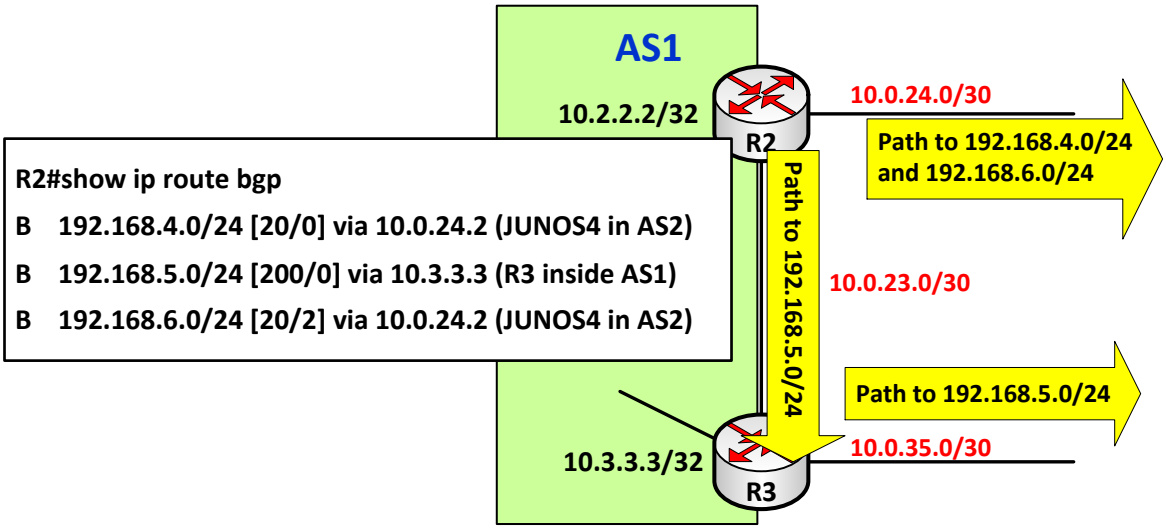


Figure 6.3

Cisco implementation of BGP has four factors to evaluate in this order:

a) the WEIGHT
b) the LOCAL-PREFERENCE
c) the AS_PATH
d) the Metric

For practical reasons, this evaluation will be explained only with the prefix 192.168.5.0/24 which is highlighted in the following snippet. There are two valid paths to reach the prefix. The best path points to R3 and the alternative path points to J4.

R2#show ip bgp

Network	Next Hop	Metric	LocPrf	Weight	Path
*> 192.168.1.0	10.0.12.1	30720		32768	i
*> 192.168.2.0	0.0.0.0	0		32768	i
*> 192.168.3.0	10.0.23.2	30720		32768	i
*> 192.168.4.0	10.0.24.2			0	2 i
***>i192.168.5.0**	**10.3.3.3**	**0**	**100**	**0**	**2 i**
*****	**10.0.24.2**	**2**		**0**	**2 i**
* i192.168.6.0	10.3.3.3	2	100	0	2 i
*>	10.0.24.2	2		0	2 i

Table 6.1

First, router R2 compares the **Cisco Weight factor** for the two entries of 192.168.5.0/24. A higher weight value is considered more preferable. But they are both zero so this element cannot be used to make a decision. The router must continue with the next step in the evaluation process.

Second, R2 evaluates the **LOCAL PREFERENCE BGP attribute**. The path via 10.3.3.3 clearly has a value of LocPrf 100 while the path via 10.24.24.2 shows nothing. Now, what does "nothing" mean? Cisco has a proviso [1] that says "...A path without LOCAL_PREF is considered to have a value of 100 by default..." According to this, the empty spaces under the LocPrf column have an implicit value of 100. (Cisco should really fix this ambiguity). When the Local Preferences of the two paths are evaluated they are found to be the same value and there is a tie again in the decision process.

Third, the **AS-PATHs** are evaluated but they have the same length, so there is no decision yet.

Finally, the **metric** is compared and a cost of zero obviously beats a cost of 2. That is why the path to route 192.168.5.0 is via R3 because the metric is lower.

The path chosen happens to be advertised using iBGP. In Cisco IOS, the Administrative Distance AD of iBGP (200) is much worse than the AD of eBGP (20). In other words, eBGP announced routes should be given preference. However, what occurs here is that the final decision over the prefixes that get installed in the IP routing table is a prerogative of the IP protocol. However, in this case, IP receives what BGP has already found to be the best path according to the BGP

algorithm. The other path, advertised via eBGP, is not presented to IP to be incorporated into the IP routing table.

So the best path has been chosen because of its metric. Now, the question is how these metrics got into the BGP database in the first place. The following diagram illustrates the situation for prefixes 192.168.4.0/24 and 192.168.5.0/24.

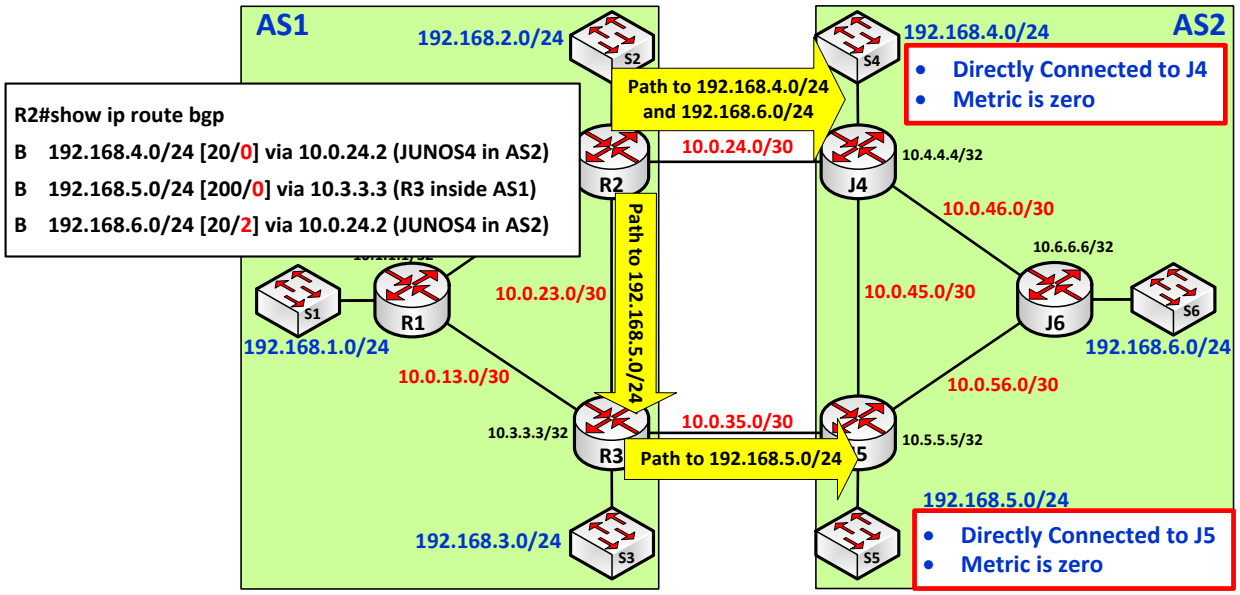


Figure 6.4

Because these networks are directly connected to border routers J4 and J5 their metrics are zero. The routing protocol BGP receives the prefixes from the IP routing table and the metrics are copied onto the MED attribute field in the UPDATE message. This is how routers R2 and R3 learn the IGP metrics.

Similarly, border routers J4 and J5 perform the metric calculation for 192.168.6.0/24 using the Shortest Path First algorithm (in OSPF) as it is shown in the diagram below. The resulting metric 2 gets installed in the IP routing tables. From there, the routers will advertise their best IP knowledge to their BGP neighbours.

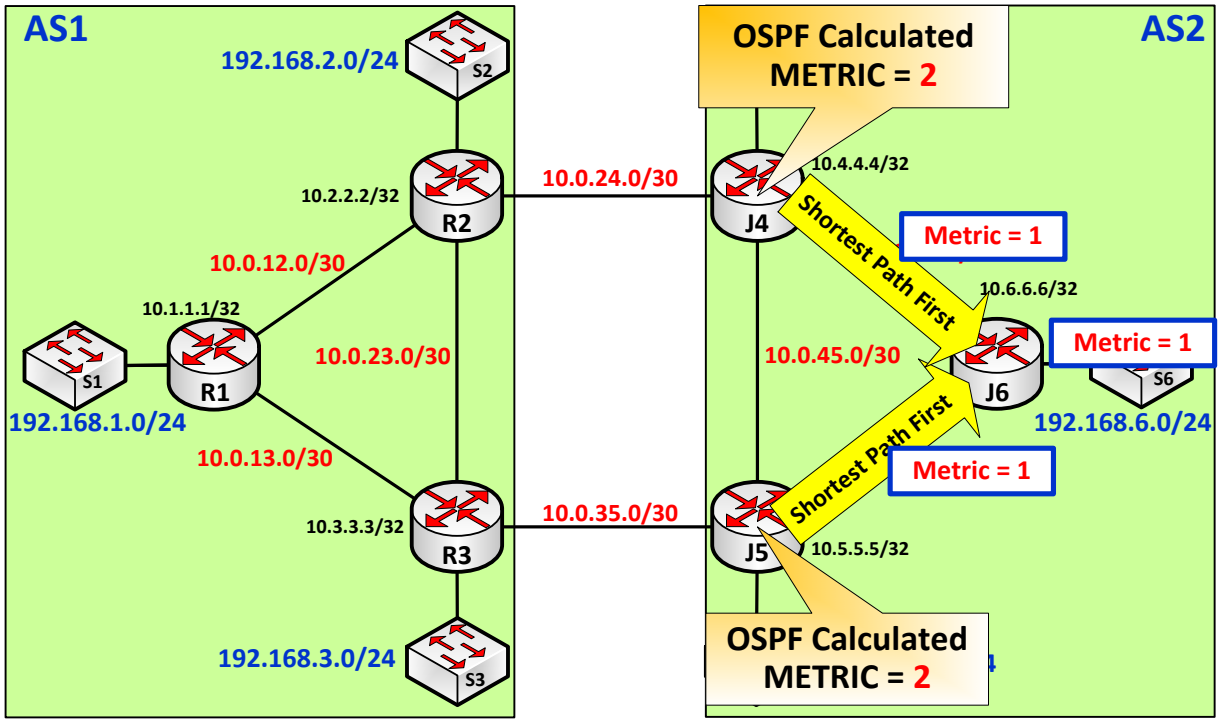


Figure 6.5

For example in the next diagram, JUNOS4 router takes the metrics from the routing table and passes those values onto the MED Attribute fields of the BGP UPDATE messages. Router R2 receives the values and it proceeds to perform the evaluation explained in the previous paragraphs.

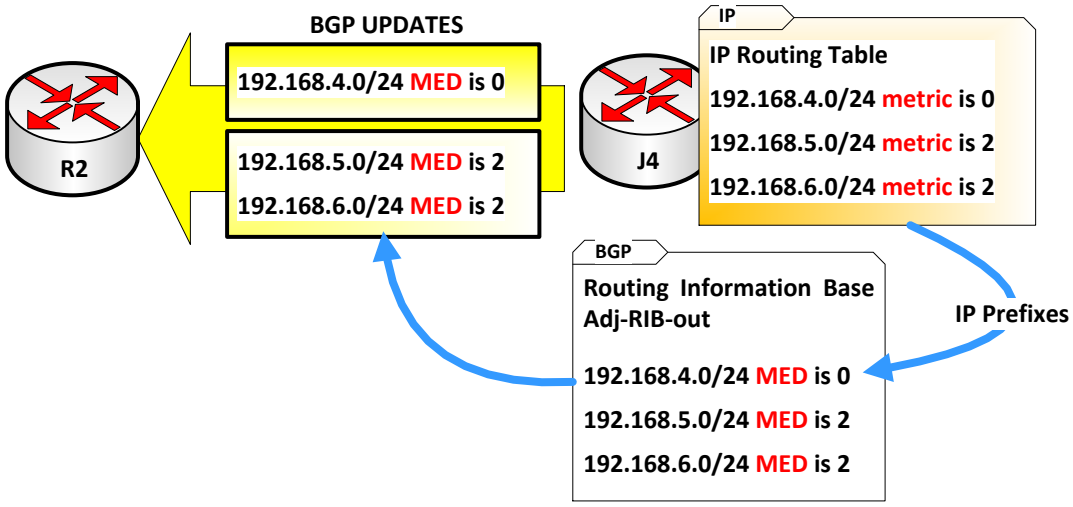


Figure 6.6

The philosophy is that the MED attribute represents by default the internal metrics of the autonomous system that generated the routing information. That makes sense to a neighbour AS that can use these values to determine the most optimum paths. The details of the internal connections are not needed; just the metric values are enough. The following diagram summarizes the network knowledge of border routers R2 and R3. Both routers have obtained the metric cost to the destinations inside the neighbour cloud from the MED values.

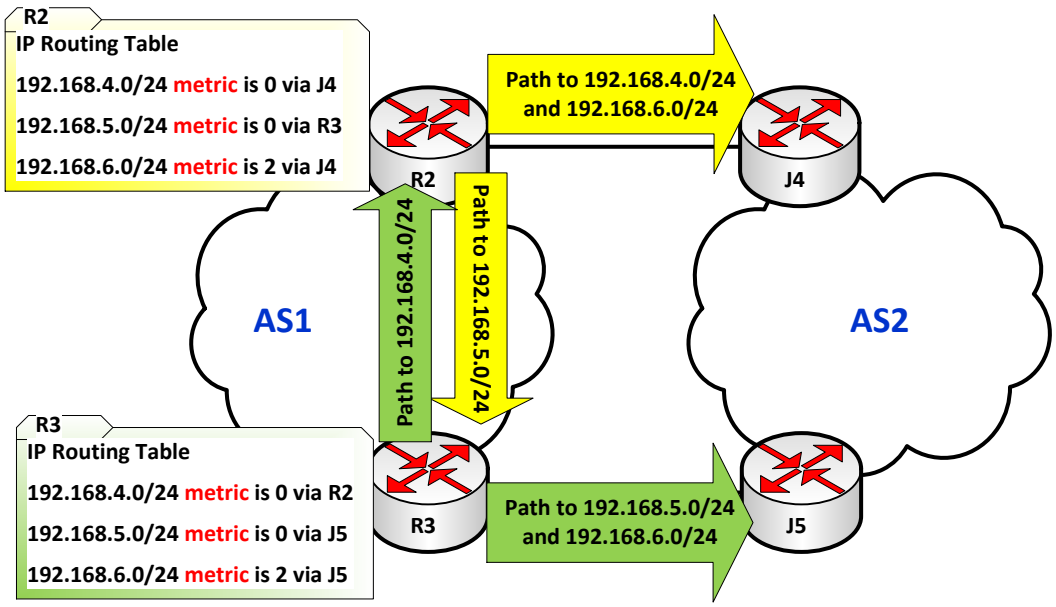
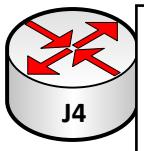


Figure 6.7

It is the turn of examining the routing knowledge of AS2. Let's remind that all the routers in AS2 are Junipers. The ASBR JUNOS4 knows the following routing information advertised by BGP:



felix@J4> show route protocol bgp terse

A	Destination	P	Preference	Metric 1	Metric 2	Next Hop	AS Path
*	192.168.1.0/24	B	170	100	30720	10.0.24.1 (R2)	1 i
		B	170	100	30720	10.0.45.2 (J5)	1 i
*	192.168.2.0/24	B	170	100	0	10.0.24.1 (R2)	1 i
*	192.168.3.0/24	B	170	100	0	10.0.45.2 (J5)	1 i
		B	170	100	30720	10.0.24.1 (R2)	1 i

Table 6.2

In JUNOS, there is only one Preference (P) value for both BGP and iBGP. Preference would be the equivalent to the Administrative Distance in Cisco [2]. However, internally, the JUNOS software process knows where the routes came from. So even though there is only one value 170 for the BGP protocol, JUNOS [2] "...always prefers eBGP routes over iBGP routes..." assuming that all other factors have been considered and found to be equal.

JUNOS has a different perspective that Cisco regarding the metrics. There are two metrics. The Metric 1 is actually the LOCAL-PREFERENCE and the default value of 100 is clearly stated for any route learned from BGP as the previous table shows. Such default value can be modified with policies. Remember from previous paragraphs that Cisco IOS presents some ambiguity by leaving some routes with empty Local-Pref values but that have an implicit meaning of 100.

Moreover, the Metric 2 represents the MED and it corresponds, by default, to the IGP metric of the source of the route. Such default value can be modified with policies. In this particular case, Metric 2 with value 30720 is the composite metric calculated by EIGRP in AS1 while the Metric 2 with zero value corresponds to a directly connected network in AS1. This situation is described in the following figure.

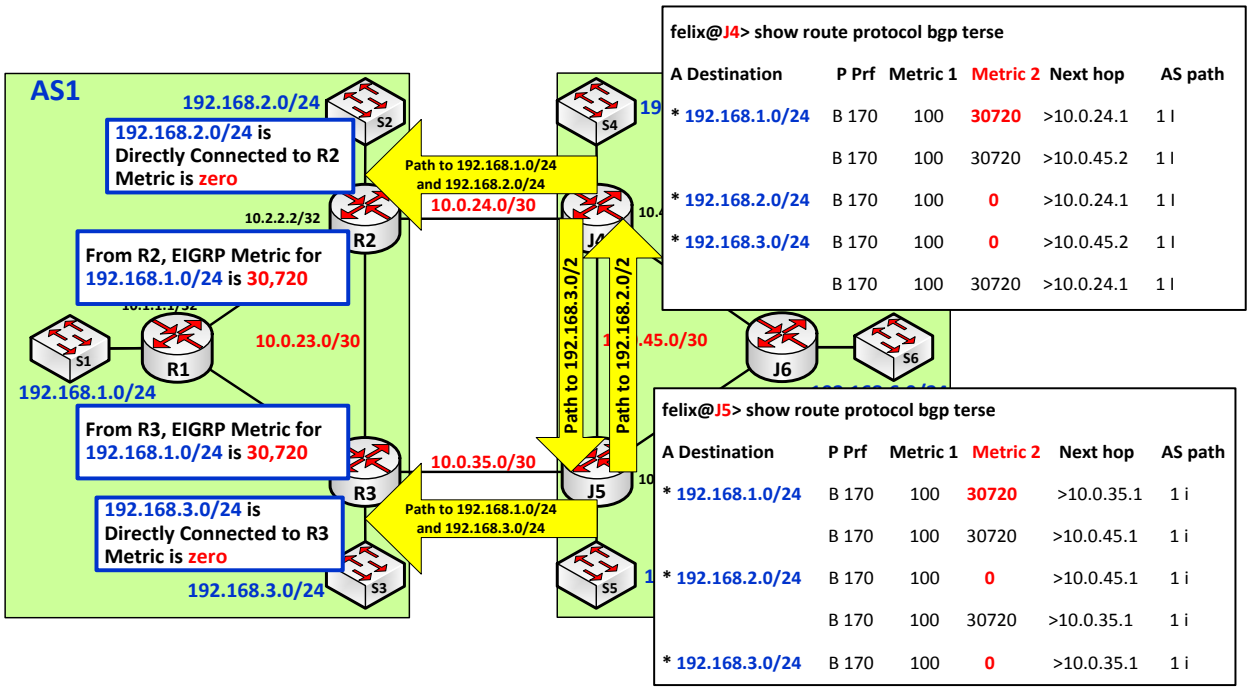


Figure 6.8

Router JUNOS4 receives from router R2 the BGP UPDATES that contain its best routing information. However, J4 determines that there is a better path to get to 192.168.3.0/24 via

JUNOS5. The metric is again the determining factor for the decision since a metric of zero wins over 30720. Notice also how the BGP UPDATES carry the metric values inside the MED fields and then JUNOS passes those values onto the Metric 2 column in the BGP routing table.

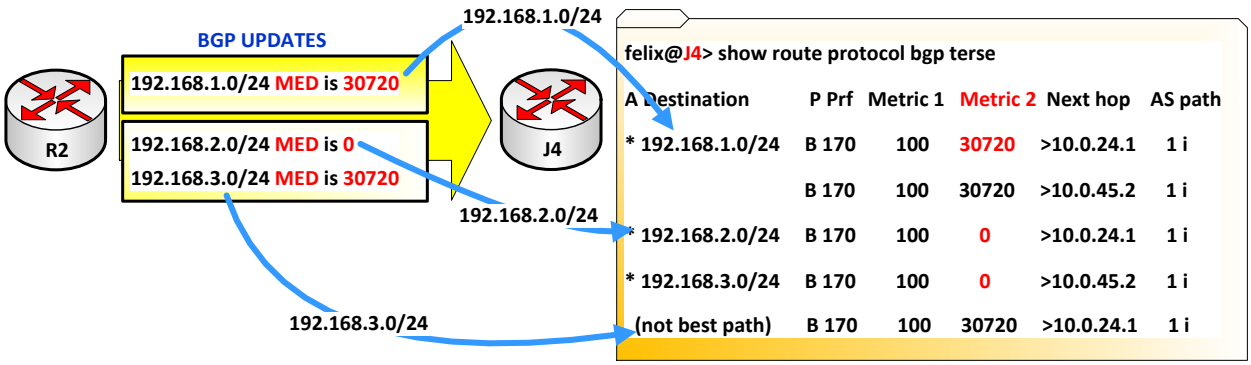
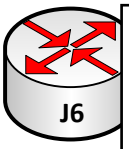


Figure 6.9

The other entries in the table are advertised by J5 via 10.45.45.2. The case of 192.168.1.0/24 is particularly interesting to analyze. Border routers Cisco R2 and R3 have calculated the same IGRP (30720) cost to reach this network attached to R1. Furthermore, R2 advertises this IGP metric to J4 while R3 does the same to J5. In turn, both Juniper routers advertise these paths to each other but the eBGP route wins over the iBGP because again, JUNOS [2] prefers eBGP routes over iBGP routes. The principle behind this implementation is to get the traffic out of the autonomous system as soon as possible.

Inside AS2, router J6 receives routing information from the border routers J4 and J5 via iBGP. The prefix 192.168.2.0/24 is advertised by J4 with a metric of zero, nothing can be beat that. The same happens to 192.168.3.0/24 with metric 0 advertised by J5. However, the situation with 192.168.1.0/24 is a mirror of what happened in AS1 with 192.168.6.0/24.



felix@J6> show route protocol bgp terse

A	Destination	P	Preference	Metric 1	Metric 2	Next Hop	AS Path
*	192.168.1.0/24	B	170	100	30720	10.0.46.1 (J4)	1 i
	192.168.1.0/24	B	170	100	30720	10.0.56.1 (J5)	1 i
*	192.168.2.0/24	B	170	100	0	10.0.46.1 (J4)	1 i
*	192.168.3.0/24	B	170	100	0	10.0.56.1 (J5)	1 i

Figure 6.10

There are two equally valid paths to the destination, so J4 goes thru the BGP decision process and after considering LOCAL-PREF and MED to be equal, it decides based on the router-id. Since J4 router-id is better than J5 that is the path chosen.

The main point of this analysis is that if BGP is left alone, it will do its calculations and make its decisions and eventually the paths to the destinations will be selected. The problem is that for the network engineer the default result might not be the desired solution. Consequently, the network administrator would like to implement policies to enforce certain traffic paths. In the following sections, the techniques for route manipulation are explained. Three implementations will be tried next to modify the path: Cisco Weight, Local Preference and Multi Exit Discriminator MED.

Using Cisco Weight to modify outbound paths

A lot of literature mistakenly claims that Cisco WEIGHT is a BGP attribute but it is not. Instead, Weight is a Cisco tool to modify paths. It is implemented to match inbound routes and it is local to the Cisco router where it is configured. **Higher values of WEIGHT imply higher preference**. Directly connected networks have a default Weight of 32,768 which is the highest possible value. Any other distant route has a default Weight value of zero.

Therefore, its application consists in changing the Weight for a path to a higher value than any other. For example, in router R1, the prefix 192.168.6.0/24 advertised by R3 will be given preference over the same prefix advertised by R2. Thus, a route map will match that condition and set a high Weight. This is the configuration:

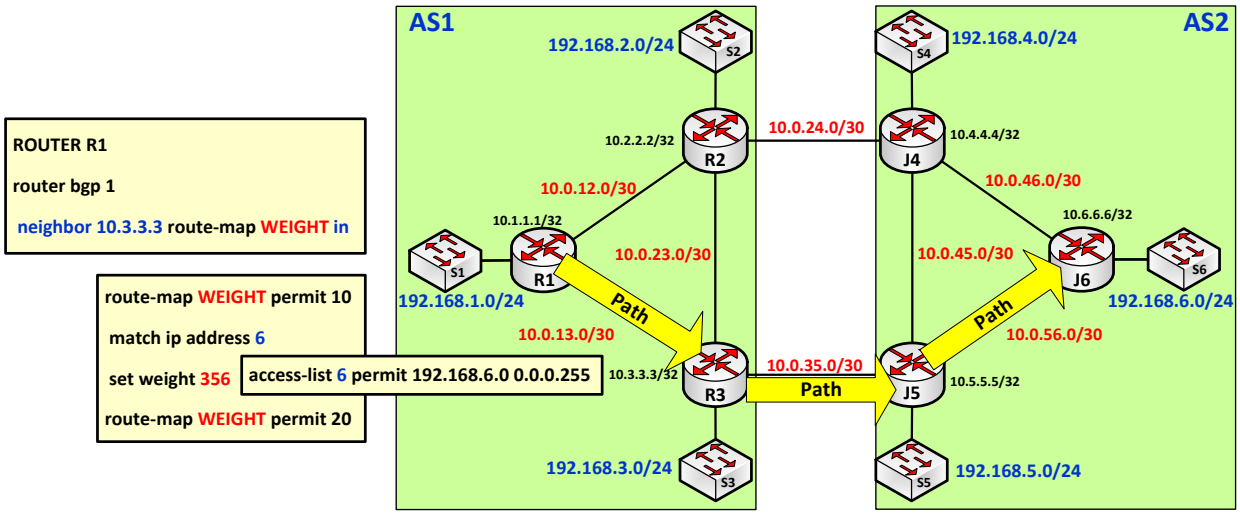


Figure 6.10

The route-map is applied in R1 to the inbound UPDATES sent by R3. Out of the routes advertised, only 192.168.6.0/24 is matched to get the weight 356. The remaining routes are allowed to go thru unchanged by the "permit 20" statement yielding these results:

R1#show ip bgp (before applying weight policy)

Network	Next Hop	Metric	LocPrf	Weight	Path
* i192.168.6.0	10.3.3.3	0	100	0	2 i
*>i	10.2.2.2	0	100	0	2 i

R1#show ip bgp (after applying weight policy)

Network	Next Hop	Metric	LocPrf	Weight	Path
*>i192.168.6.0	10.3.3.3	0	100	356	2 i
* i	10.2.2.2	0	100	0	2 i

Table 6.3

Before the policy, the best path to reach 192.168.6.0/24 from R1 was via router R2. Afterwards, the Weight is set to 356 and the best path has changed to go via router R3. Cisco Weight is a

simple and effective way to manipulate traffic flows in certain topologies. Let's try another technique to control the way out of the autonomous system AS1.

Using Local Preference Attribute to modify paths

Cisco Configuration

Local Preference is a BGP attribute that is used to control the exit way of traffic in an autonomous system. It is applied internally in the domain so it only has meaning inside the AS. For example, the following policy is applied in router R1, inbound, to match the advertisement of 192.168.6.0/24 from R3 and to set the local preference value to 356.

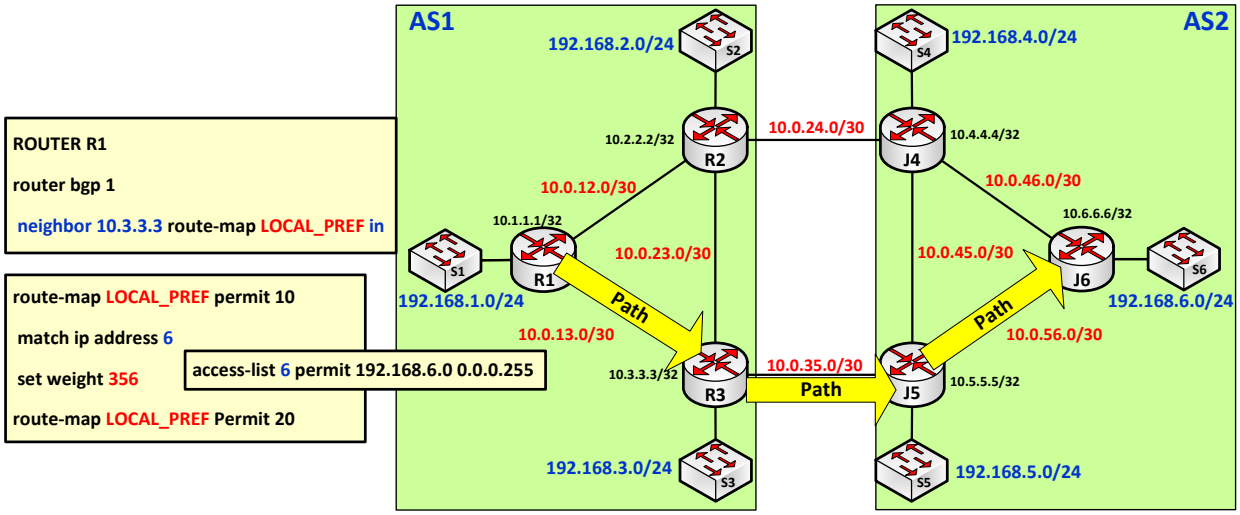


Figure 6.11

In effect, since 356 is a higher than the other value available (100 via R2) that causes the path via R3 to be preferred. This is the proof that the policy is working:

R1# show ip bgp

Network	Next Hop	Metric	Local Preference	Weight	Path
* i192.168.6.0	10.2.2.2	0	100	0	2 i
*>i Best path	10.3.3.3	0	356	0	2 i

Table 6.4

Alternately, because Local Preference is a BGP attribute and it is carried inside the UPDATE message, then it can be also modified at the border router R3 to cause the same effect in R1. For example:

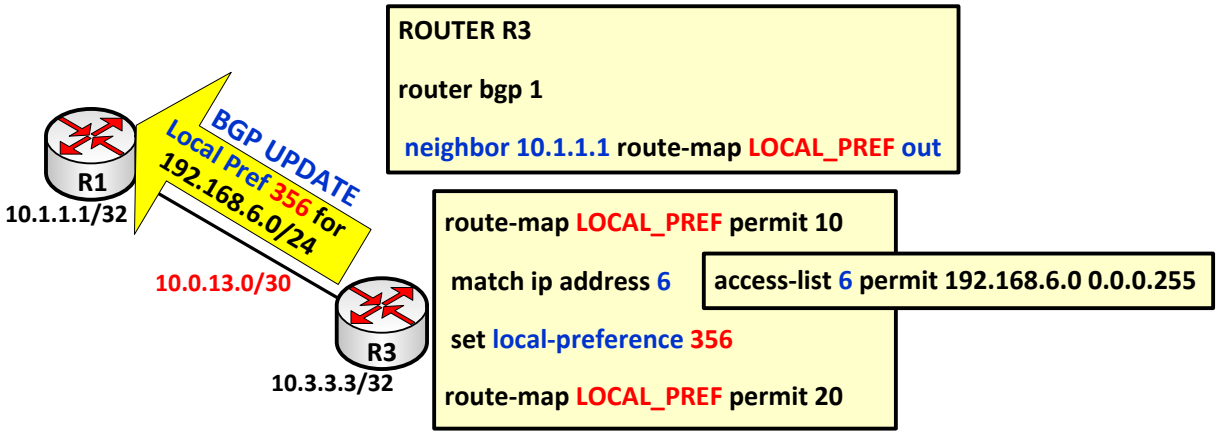


Figure 6.12

The result is the same as in the previous case but now the route map is implemented in router R3 for the updates going outbound R1. The IP network 192.168.6.0/24 gets its Local Preference Attribute set to 356. When R1 receives the routing information, it proceeds to install that in the BGP database. A higher Local Preference makes the path toward R3 more appealing.

Juniper Configuration

The LOCAL PREFERENCE attribute value can be modified in different ways in JUNOS. In this example, the attribute is changed in router J4 to a lower value so the other routers would not use J4 for a best path to the destination 192.168.1.0/24 in AS1. In the iBGP configuration for AS2, there is already a policy to accept the prefixes advertised by AS1 only. This policy is modified to set the local-preference to 50 which is lower than the default preference.

```
Router JUNIPER J4
set protocols bgp group iBGP type internal
set protocols bgp group iBGP local-address 10.4.4.4
set protocols bgp group iBGP local-as 2
set protocols bgp group iBGP neighbor 10.5.5.5 peer-as 2
set protocols bgp group iBGP neighbor 10.6.6.6 peer-as 2
set protocols bgp group iBGP export iBGP
```

This policy

```
set policy-options policy-statement iBGP term 1 from protocol bgp
set policy-options policy-statement iBGP term 1 from route-filter 192.168.1.0/24 exact
set policy-options policy-statement iBGP term 1 then local-preference 50
set policy-options policy-statement iBGP term 1 then next-hop self
set policy-options policy-statement iBGP term 1 then accept

set policy-options policy-statement iBGP term 2 from protocol bgp
set policy-options policy-statement iBGP term 2 from route-filter 192.168.2.0/24 exact
set policy-options policy-statement iBGP term 2 then next-hop self
set policy-options policy-statement iBGP term 2 then accept

set policy-options policy-statement iBGP term 3 from protocol bgp
set policy-options policy-statement iBGP term 3 from route-filter 192.168.3.0/24 exact
set policy-options policy-statement iBGP term 3 then next-hop self
set policy-options policy-statement iBGP term 3 then accept

set policy-options policy-statement iBGP then reject
```

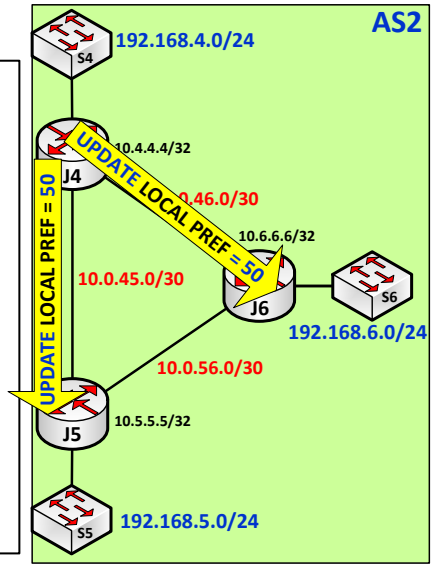


Figure 6.13

Router J4 advertises the prefix 192.168.1.0/24 with a local-preference of 50 to both J5 and J6. The routers will compare this entry with the other path via J5 which offers a higher preference of 100. The result is that the route will not be used as a best path to the destination. This effect can be observed in JUNOS6 in the next figure.

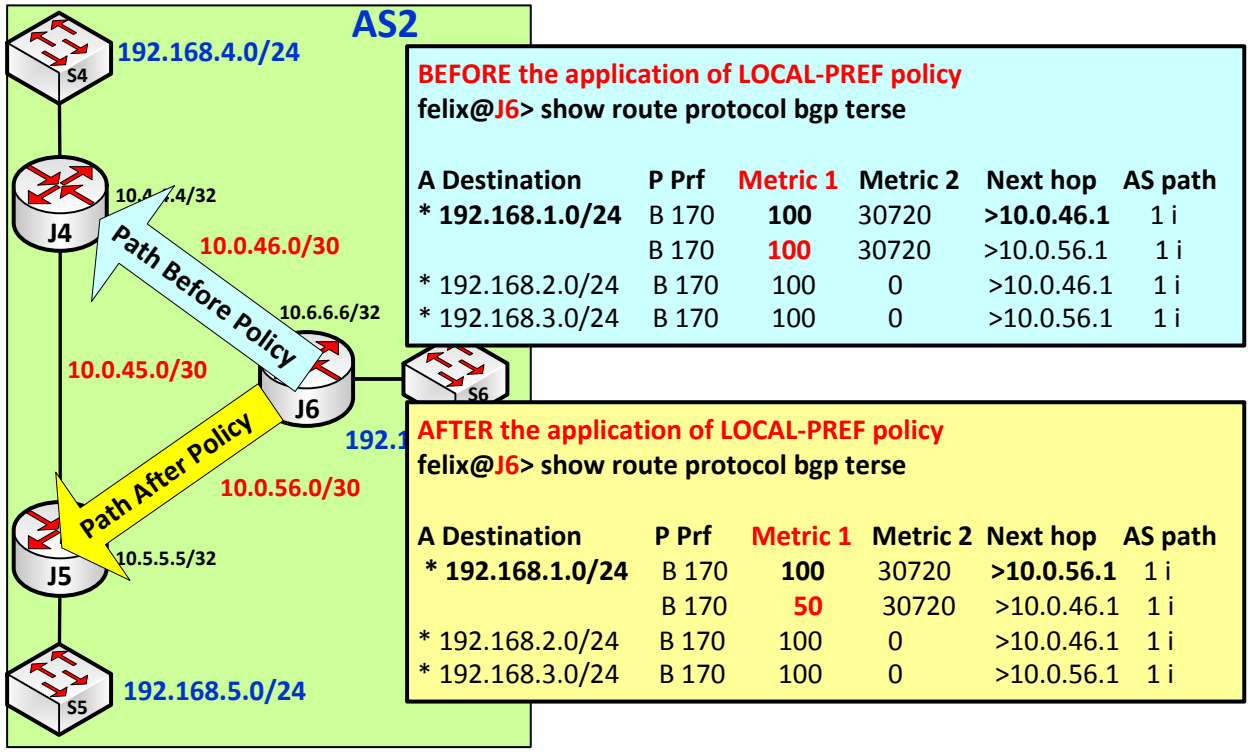


Figure 6.14

Notice how the Metric 1 has changed from the default value of 100 to 50 and the best path (*) has switched from 10.0.46.1 to 10.0.56.1. The other routes remain with the default Local-Preference unchanged.

The manipulation of the Local Preference Attribute is typically done to affect the routing knowledge of routers located inside the autonomous systems and it can be implemented either locally in the internal router or it can be implemented at the border routers to advertise the attribute via iBGP to the internal routers.

Using Multi-Exit-Discriminator to modify paths

Cisco implementation

The administrator of AS1 can change the metrics of the routes being advertised to AS2 to make some paths more desirable than others. For example, to reach 192.168.1.0/24 from AS2, the path between R3 and J5 could be preferred instead of the link between R2 and J4. To cause this effect, a larger metric is advertised from R2 toward J4 and a lower metric is advertised by R3 to J5. Obviously, a lower metric is better than a larger metric. This is the implementation in the Cisco border routers R4 and R5.

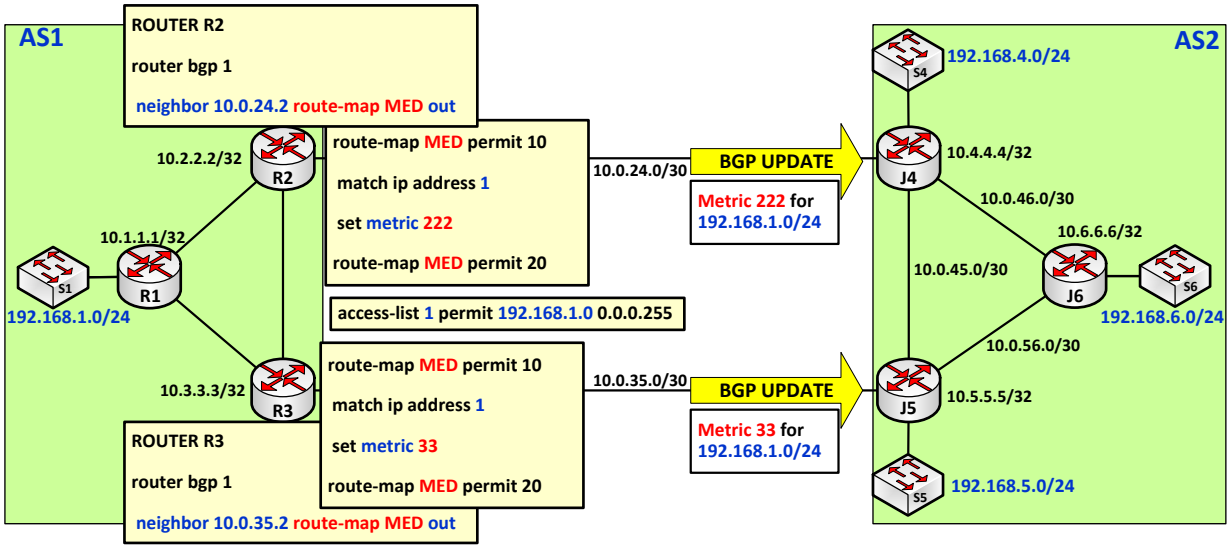


Figure 6.15

Actually, the policy implemented in R3 alone would be enough to have the desired effect since the EIGRP metric is so high (30720) anyway. However, the best practice is to enforce the policies regardless of the default status.

This is the result in Juniper J4.

BEFORE

felix@J4> show route protocol bgp terse

A Destination	P Prf	Metric 1	Metric 2	Next hop	AS path
* 192.168.1.0/24	B 170	100	30720	>10.0.24.1	1 i
	B 170	100	30720	>10.0.45.2	1 i
* 192.168.2.0/24	B 170	100	0	>10.0.24.1	1 i
* 192.168.3.0/24	B 170	100	0	>10.0.45.2	1 i
	B 170	100	30720	>10.0.24.1	1 i

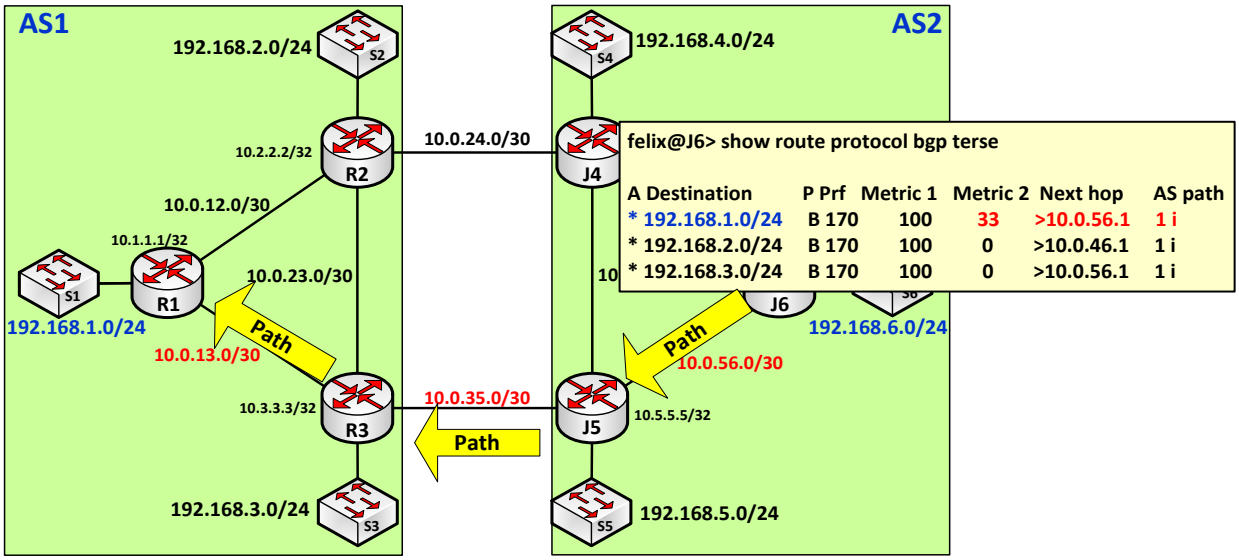
AFTER

felix@J4> show route protocol bgp terse

A Destination	P Prf	Metric 1	Metric 2	Next hop	AS path
* 192.168.1.0/24	B 170	100	33	>10.0.45.2	1 i
	B 170	100	222	>10.0.24.1	1 i
* 192.168.2.0/24	B 170	100	0	>10.0.24.1	1 i
* 192.168.3.0/24	B 170	100	0	>10.0.45.2	1 i
	B 170	100	30720	>10.0.24.1	1 i

Table 6.5

The Metric 2 contains the MED values advertised using BGP. It is important to notice that the router has received two entries for the same route 192.168.1.0/24. One entry has the higher metric 222 coming from the direct neighbour R4-AS1 while the other entry has the lower metric 33 coming from the internal peer JUNOS5. That proves that iBGP carries the MED value as well as eBGP. The consequence is that the lower MED 33 forces the best path to go toward JUNOS5. This is the result in JUNOS6, only the path with the lowest MED appears in the routing table.



felix@J6> show route protocol bgp terse

A Destination	P Prf	Metric 1	Metric 2	Next hop	AS path
* 192.168.1.0/24	B 170	100	33	>10.0.56.1	1 i
* 192.168.2.0/24	B 170	100	0	>10.0.46.1	1 i
* 192.168.3.0/24	B 170	100	0	>10.0.56.1	1 i

Figure 6.16

Juniper implementation

The previous policy is removed and a mirror implementation of MED manipulation is now done in AS2 with the Juniper router J4. The effect of it will be observed in Cisco router R2 in AS1. Route 192.168.6.0/24 is assigned the MED 66 by the policy named To-AS1. All the routes being advertised by J4 are controlled by such policy. Observe that the third statement (term 3) accepts the route 192.168.6.0/24 and then it changes the metric to the value 66.

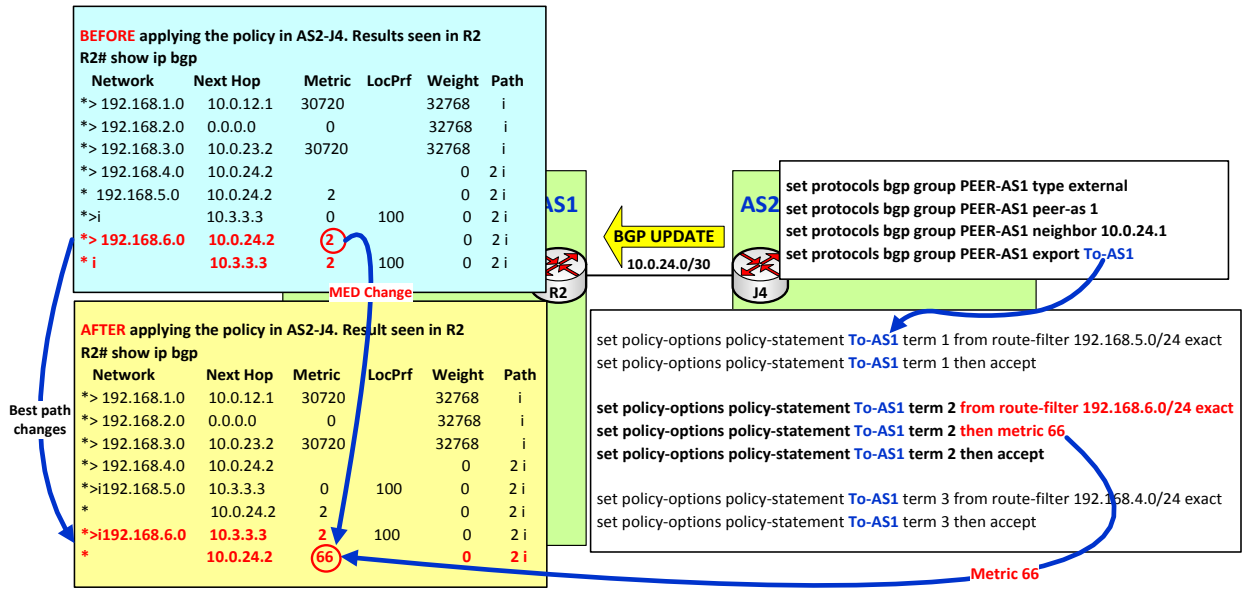


Figure 6.17

When R2 receives the update it proceeds to change the previous Metric 2 value to 66 making the route less appealing. The situation before applying the policy and after its application is illustrated in the figure. The best path has changed from 10.0.24.2 to 10.3.3.3 while the other routes remain unchanged.

Using AS-PATH to change path appeal

Another technique to manipulate traffic paths is to artificially add autonomous system numbers to the AS-PATH attribute. The idea is to make the longer paths less appealing to the BGP decision process of choosing the best path. This manipulation technique is called **AS-Prepending**.

By default, any AS border router stamps its own AS number to the UPDATES of prefixes sent out to an external peer. This AS number appears in the AS-PATH Attribute field and it accumulates the ASNs that are in the path to the destination. AS-Prepending consists on adding more ASNs to the default ASNs already present in the AS-PATH attribute.

Cisco implementation

Router Cisco R2 is applying AS-prepending toward J4 in AS2. A route map matches the prefix 192.168.1.0/24 and then adds the ASN twice. This makes a total AS-PATH length of three AS1 including the default ASN. The result is observed in AS2 with the longer, less desirable path.

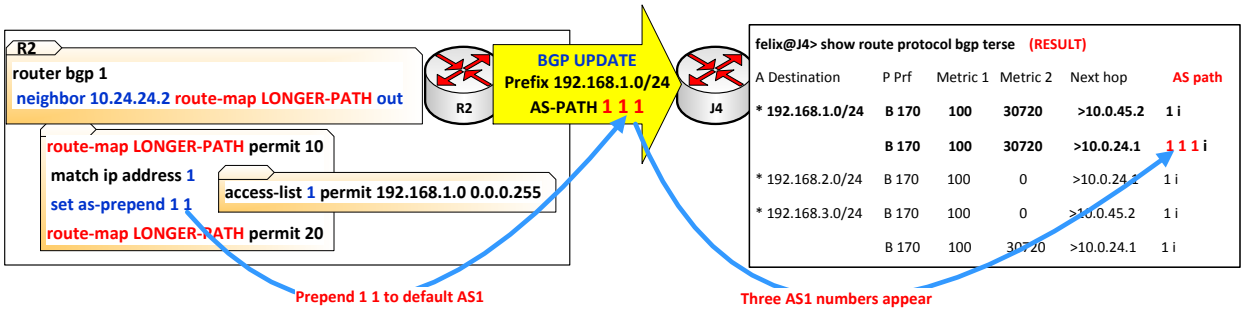


Figure 6.18

An interpretation of J4 routing knowledge for 192.168.1.0/24 is presented in the next figure. Prepending leads J4 to

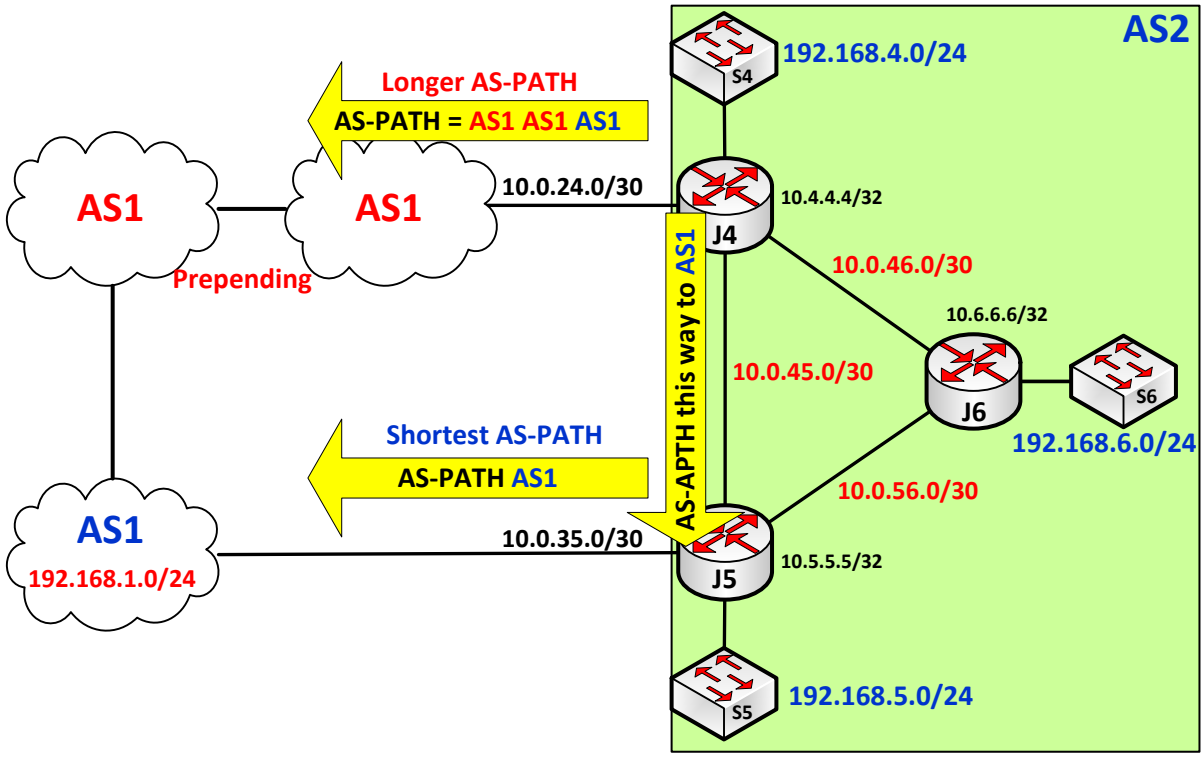


Figure 6.19

Juniper implementation

The export policy To-AS1 in J4 is modified to include the "as-path-prepend" condition. Different amount of prepending is applied to AS2 routes to better show the effect of the AS-PATH manipulation. For example, prefix 192.168.6.0/24 gets three ASN (2 2 2) that will result in an AS-PATH length of four ASN including the AS added by the border router by default. The BGP database in R2 has been edited to show only the relevant details.

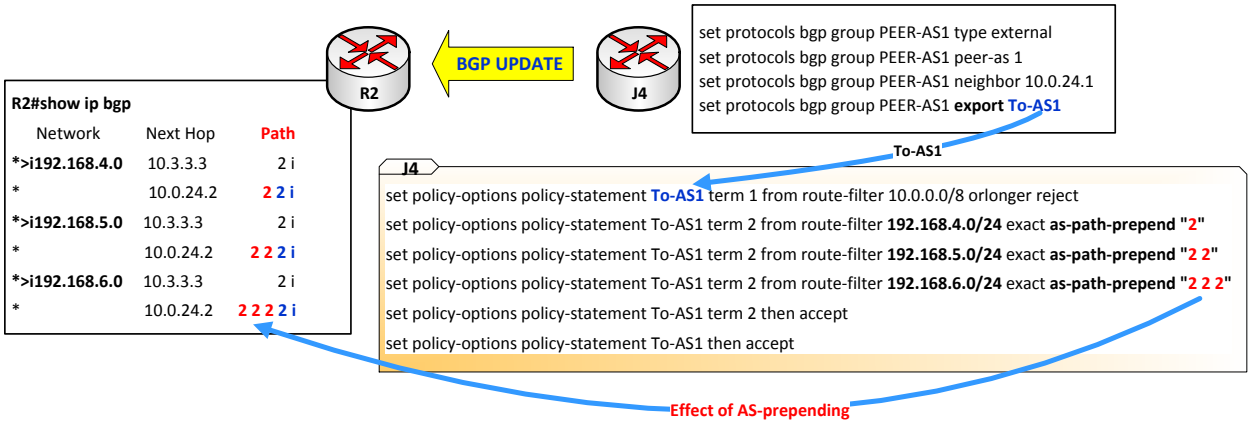


Figure 6.20

The AS prepending technique allows the inclusion of other AS numbers in the path. For instance, in the next case, private AS numbers are added to the AS-PATH.

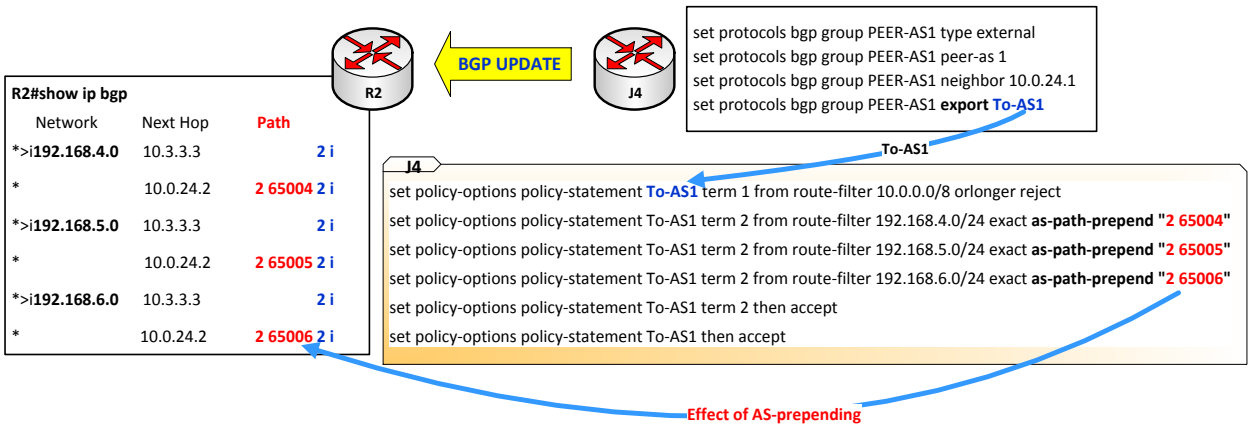


Figure 6.21

AS prepending is another tool available for path manipulation but it must be done carefully following best practices. The prepending must be done with the owned autonomous system number. When it is done improperly, AS-prepending leads to confusion and routing mistakes.

Chapter Summary

The most widespread task for BGP Network engineers is the manipulation of traffic paths and that is accomplished by implementing policies that make use of the BGP attributes that precisely relate to paths values and lengths.

BGP has its own internal algorithm that, if left alone, will choose one path to a network destination. The algorithm gives preference in this order to the attributes LOCAL_PREFERENCE, AS_PATH lengths, and MED. If the router is Cisco then a WEIGHT proprietary factor precedes all the mentioned attributes. (WEIGHT is not a BGP attribute).

The BGP attribute LOCAL_PREF is used to increase the preference to a particular pathway inside an Autonomous System. The higher the number assigned to the attribute, the higher the preference level.

The AS_PATH is a mandatory attribute with the basic function of showing the length in AS numbers to a network destination. However, its value can be manually modified by the network administration to make it longer or less appealing.

Multi-Exit-Discriminator or MED based policies can implemented when peering autonomous systems have two or more direct paths between them. MED is fundamentally a metric value, so the higher the value, the less desirable. The use of MED is only meaningful between two AS neighbours since the attribute is not transitive.

Exercises

This network topology will be used for the policies specifies below.

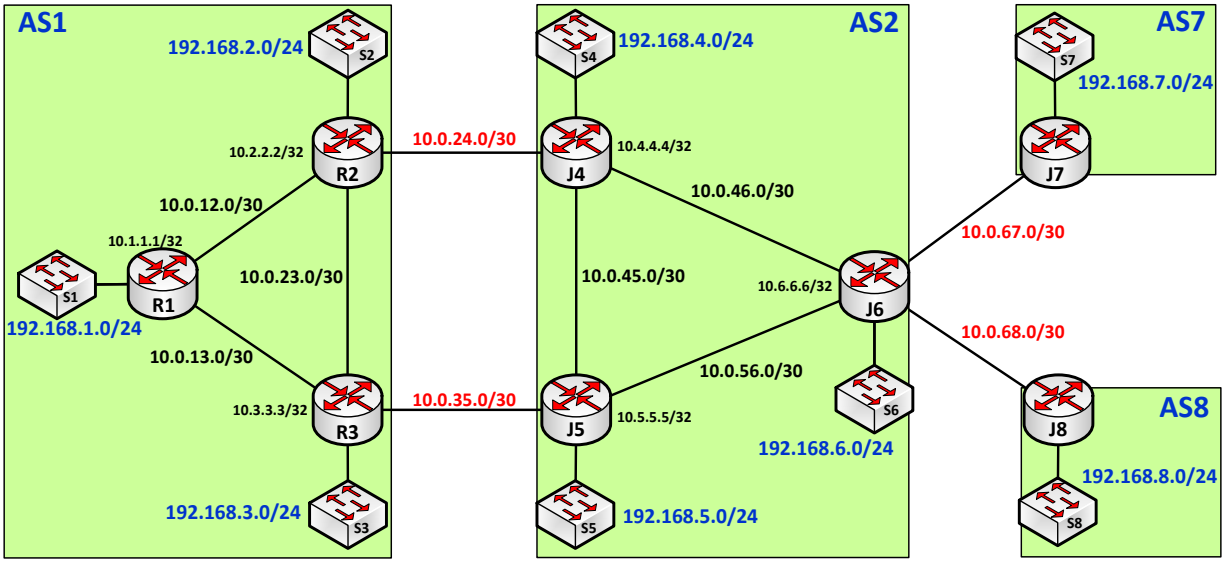


Figure 6.22

Implement a policy to make the traffic flowing from source network 192.168.1.0/24 to destination network 192.168.8.0/24 to follow this path:

- From R1, then R3, then R5, R6 and finally R8.
- Repeat the same policy but using different techniques such as: a) Cisco Weight (if R1 is Cisco), b) LOCAL-PREF, c) MED, d) AS-PATH prepending.

Implement a policy that can do all of the following:

- Traffic from 192.168.1.0/24 to 192.168.7.0/24 follows path R1, R3, R5, R6 and R7 but the return traffic follows R7, R6, R4, R2, and R1.
- Traffic between 192.168.8.0/24 and 192.168.3.0/24 follows R8, R6, R4, R2, and then R3. The return path is not controlled.
- Any other traffic flow is by default.

In AS2-J4 insert prepending of AS 2 1.

- Analyze the result by sniffing the BGP UPDATEs between J4 and R2 and by debugging the BGP messages in R2. What is happening? Explain based in the functioning of BGP.

In AS2-J4 insert prepending of AS 3 2.

- Analyze the result by sniffing the BGP UPDATEs between J4 and R2 and by debugging the BGP messages in R2. What is happening? Explain based in the functioning of BGP.

6.2) Configure this topology and implement the policies below:

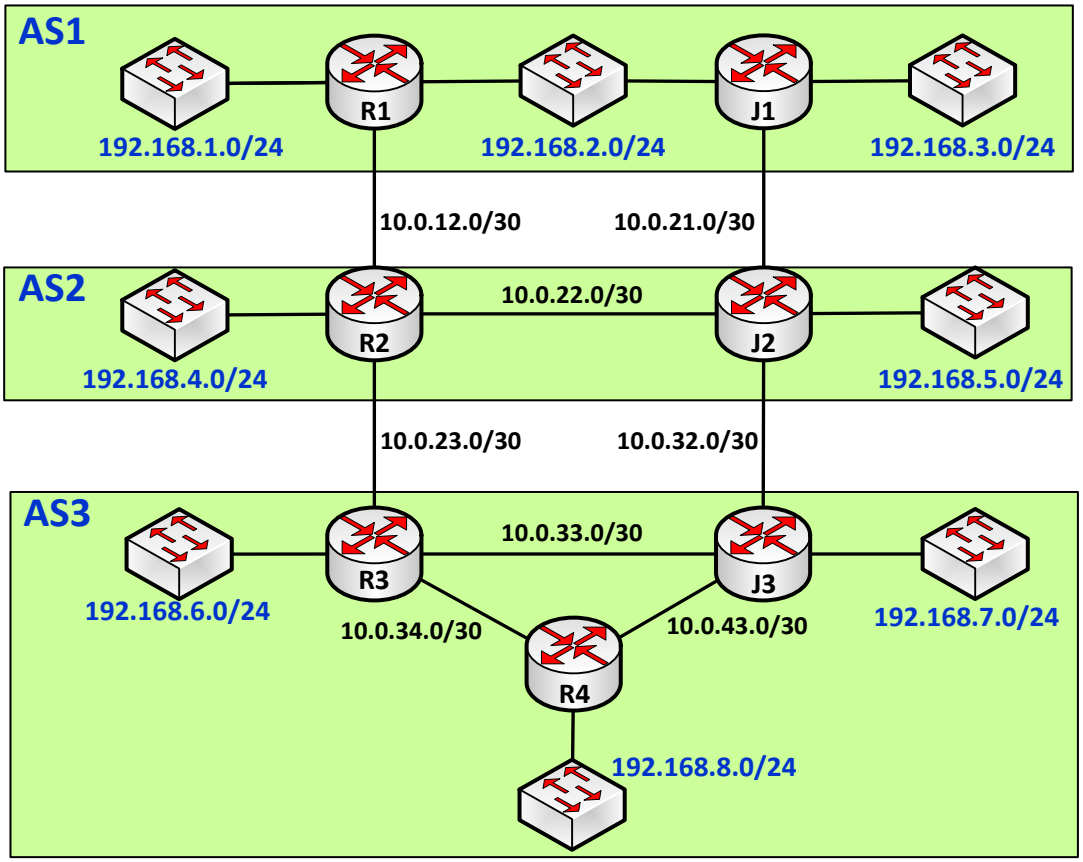


Figure 6.23

- Implement policies that will control the traffic among the IP networks using a combination of MED and LOCAL_PREF.
- Traffic from S6 to S1 should flow via next hop 10.23.23.1
- Traffic from S6 to S3 should flow via 10.32.32.1
- Traffic from S1 to S5 should flow via 10.21.21.2
- Traffic from S3 to S4 should flow via 10.12.12.2

6.3) Configure the following topology and then implement the policies to control traffic flows as indicated below the diagram.

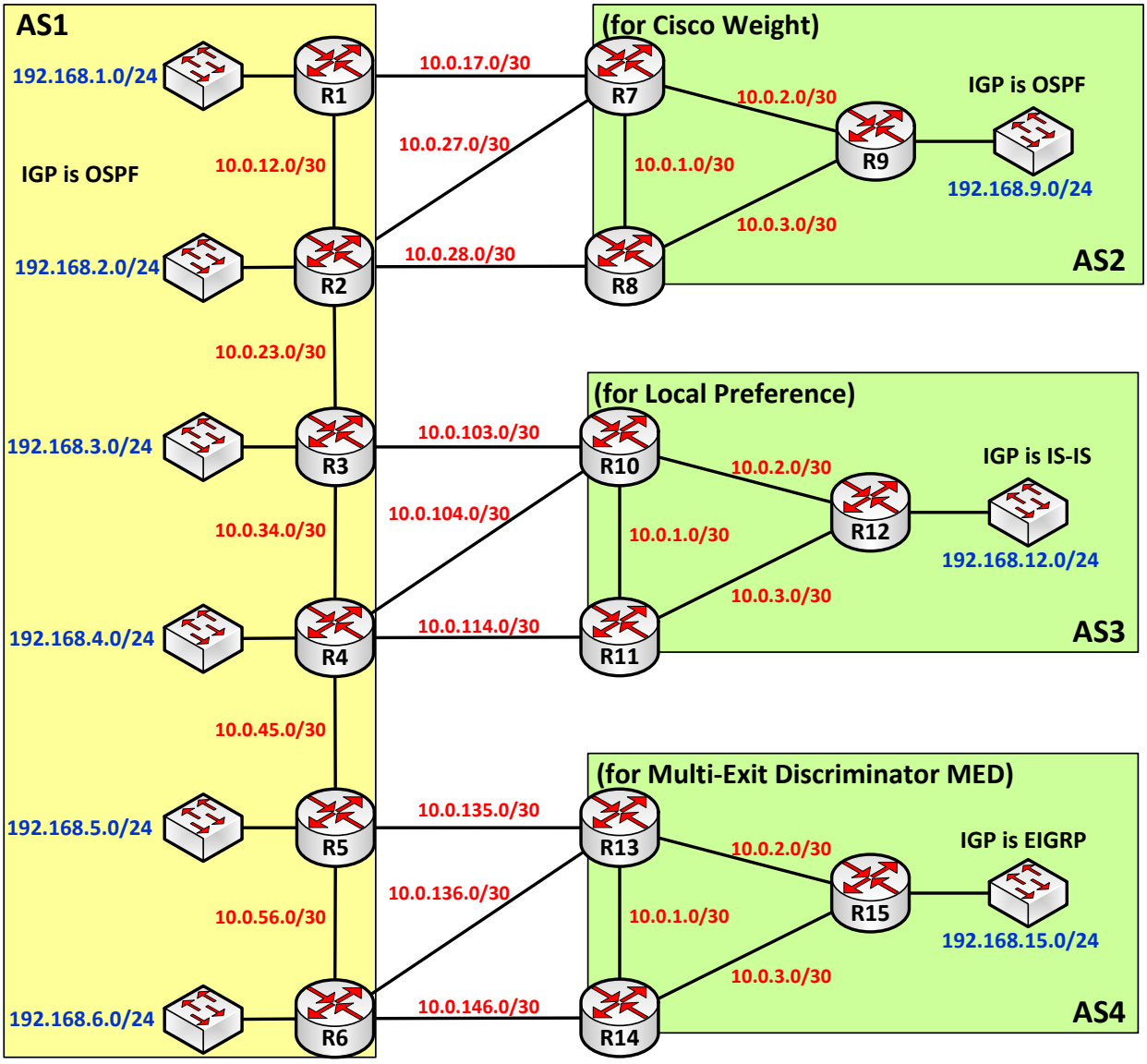


Figure 6.24

In AS2, collect the (no-policy) IP routing and BGP tables, and analyze the paths obtained by default.

In AS2, implement one policy that uses Cisco Weight to achieve the following:

- R9 accesses 192.168.2.0/24 via R7, then R7 accesses the same prefix via R1.
- R9 accesses 192.168.1.0/24 via R8, then R8 accesses the same prefix via R7.

In AS3, collect the (no-policy) IP routing and BGP tables, and analyze the paths obtained by default.

In AS3, implement one policy that uses LOCAL_PREF achieve the following:

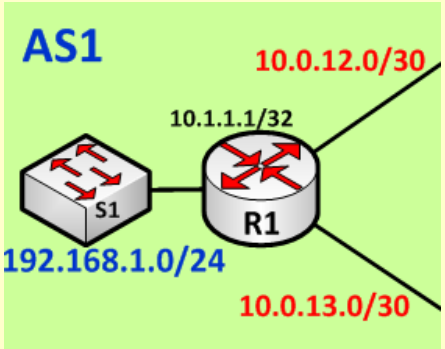
- R12 prefers to access IP routes 192.168.1.0/24, 192.168.2.0/24, and 192.168.3.0/24 via R11.
- Trace route this policy and follow the path, explaining every hop with the help of the BGP database tables.

In AS4, implement one policy that uses MED to achieve the following:

- Router R5 should be the path that R13 uses to reach 192.168.6.0/24.
- R13 should be the path for R5 to reach 192.168.15.0/24.
- R14 should be the path for R6 to reach 192.168.15.0/24.

Appendix

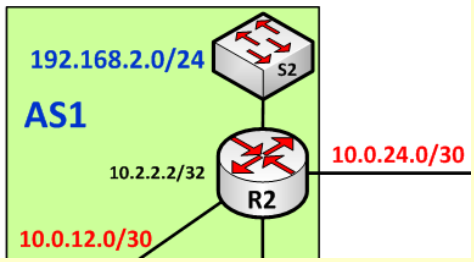
```
hostname R1
!
```




```
interface Loopback0
 ip address 10.1.1.1 255.255.255.255
!
interface FastEthernet0/0
 ip address 192.168.1.1 255.255.255.0
no shutdown
 !
interface FastEthernet1/0
 ip address 10.0.12.1 255.255.255.252
no shutdown
!
interface FastEthernet1/1
 ip address 10.0.13.1 255.255.255.252
no shutdown
```

```
hostname R1
!
router eigrp 1
 network 10.1.1.1 0.0.0.0
 network 10.0.12.0 0.0.0.3
 network 10.0.13.0 0.0.0.3
 network 192.168.1.0
 no auto-summary
!
router bgp 1
neighbor 10.2.2.2 remote-as 1
neighbor 10.2.2.2 update-source Lo0
 neighbor 10.3.3.3 remote-as 1
 neighbor 10.3.3.3 update-source Lo0
!
```

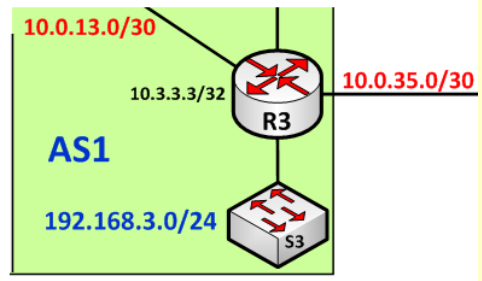
```
hostname R2
```




```
!
interface Loopback0
 ip address 10.2.2.2 255.255.255.255
!
interface FastEthernet0/0
 ip address 192.168.2.1 255.255.255.0
no shutdown
!
interface FastEthernet0/1
 ip address 10.0.24.1 255.255.255.252
no shutdown
!
interface FastEthernet1/0
 ip address 10.0.12.2 255.255.255.252
no shutdown
!
interface FastEthernet1/1
 ip address 10.0.23.1 255.255.255.252
no shutdown
!
router eigrp 1
 network 10.2.2.2 0.0.0.0
 network 10.0.12.0 0.0.0.3
 network 10.0.23.0 0.0.0.3
 network 192.168.2.0
 no auto-summary
!
```

```
hostname R2
!
router bgp 1
 network 192.168.1.0
 network 192.168.2.0
 network 192.168.3.0
 neighbor 10.1.1.1 remote-as 1
 neighbor 10.1.1.1 update-source Lo0
 neighbor 10.1.1.1 next-hop-self
 neighbor 10.1.1.1 route-map iBGP out
 neighbor 10.3.3.3 remote-as 1
 neighbor 10.3.3.3 update-source Lo0
 neighbor 10.3.3.3 next-hop-self
 neighbor 10.3.3.3 route-map iBGP out
 neighbor 10.0.24.2 remote-as 2
!
route-map iBGP deny 10
 match as-path 1
route-map iBGP permit 20
!
ip as-path access-list 1 permit ^$
```

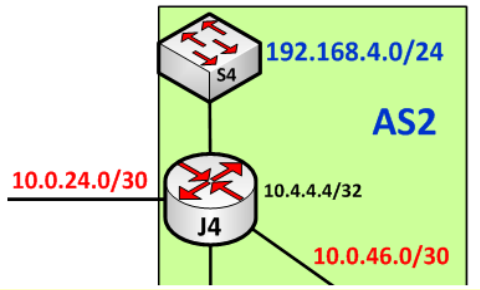
```
hostname R3
```




```
!
interface Loopback0
 ip address 10.3.3.3 255.255.255.255
!
interface FastEthernet0/0
 ip address 192.168.3.1 255.255.255.0
no shutdown
!
interface FastEthernet0/1
 ip address 10.0.35.1 255.255.255.252
no shutdown
!
interface FastEthernet1/0
 ip address 10.0.23.2 255.255.255.252
no shutdown
!
interface FastEthernet1/1
 ip address 10.0.13.2 255.255.255.252
no shutdown
!
router eigrp 1
 network 10.3.3.3 0.0.0.0
 network 10.0.13.0 0.0.0.3
 network 10.0.23.0 0.0.0.3
 network 192.168.3.0
 no auto-summary
!
```

```
hostname R3
!
router bgp 1
network 192.168.1.0
 network 192.168.2.0
 network 192.168.3.0
 neighbor 10.1.1.1 remote-as 1
 neighbor 10.1.1.1 update-source Lo0
 neighbor 10.1.1.1 next-hop-self
 neighbor 10.1.1.1 route-map iBGP out
 neighbor 10.2.2.2 remote-as 1
 neighbor 10.2.2.2 update-source Lo0
 neighbor 10.2.2.2 next-hop-self
 neighbor 10.2.2.2 route-map iBGP out
 neighbor 10.0.35.2 remote-as 2
!
route-map iBGP deny 10
 match as-path 1
route-map iBGP permit 20
!
ip as-path access-list 1 permit ^$
```

set system host-name J4



set **interfaces** em0 unit 0 family inet address 192.168.4.1/24
set **interfaces** em1 unit 0 family inet address 10.0.24.2/30
set **interfaces** em4 unit 0 family inet address 10.0.46.1/30
set **interfaces** em5 unit 0 family inet address 10.0.45.1/30
set **interfaces** lo0 unit 0 family inet address 10.4.4.4/32

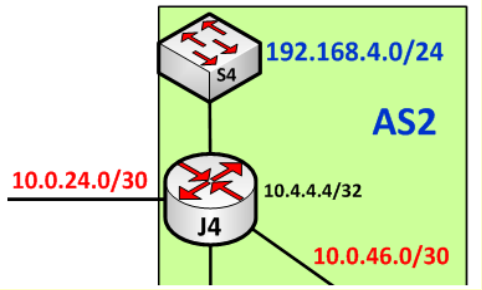
set protocols **ospf** area 0.0.0.0 interface lo0.0
set protocols **ospf** area 0.0.0.0 interface em0.0
set protocols **ospf** area 0.0.0.0 interface em5.0
set protocols **ospf** area 0.0.0.0 interface em4.0

set **routing-options** router-id 10.4.4.4
set **routing-options** autonomous-system 2

set protocols **bgp** group **PEER-AS1** type external
set protocols **bgp** group **PEER-AS1** export **To-AS1**
set protocols **bgp** group **PEER-AS1** peer-as 1
set protocols **bgp** group **PEER-AS1**1 neighbor 10.0.24.1

CONTINUES NEXT ...

set system host-name J4

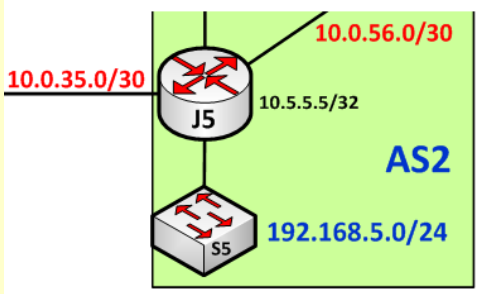


set protocols **bgp** group **INTERNAL** type internal
set protocols **bgp** group **INTERNAL** local-address 10.4.4.4
set protocols **bgp** group **INTERNAL** export **iBGP**
set protocols **bgp** group **INTERNAL** local-as 2
set protocols **bgp** group **INTERNAL** neighbor 10.5.5.5 peer-as 2
set protocols **bgp** group **INTERNAL** neighbor 10.6.6.6 peer-as 2

set **policy**-options policy-statement **To-AS1** term 1 from route-filter 192.168.5.0/24 exact
set **policy**-options policy-statement **To-AS1** term 1 then accept
set **policy**-options policy-statement **To-AS1** term 2 from route-filter 192.168.6.0/24 exact
set **policy**-options policy-statement **To-AS1** term 2 then accept
set **policy**-options policy-statement **To-AS1** term 3 from route-filter 192.168.4.0/24 exact
set **policy**-options policy-statement **To-AS1** term 3 then accept

set **policy**-options policy-statement **iBGP** term 1 from protocol bgp
set **policy**-options policy-statement **iBGP** term 1 from as-path **LOCAL**
set **policy**-options policy-statement **iBGP** term 1 then reject
set **policy**-options policy-statement **iBGP** term 2 from protocol bgp
set **policy**-options policy-statement **iBGP** term 2 from as-path **EXTERIOR**
set **policy**-options policy-statement **iBGP** term 2 then next-hop self
set **policy**-options policy-statement **iBGP** term 2 then accept
set **policy**-options as-path **LOCAL "()"**
set **policy**-options as-path **EXTERIOR .***

set system host-name J5



set **interfaces** em0 unit 0 family inet address 192.168.5.1/24
set **interfaces** em1 unit 0 family inet address 10.0.35.2/30
set **interfaces** em3 unit 0 family inet address 10.0.56.1/30
set **interfaces** em5 unit 0 family inet address 10.0.45.2/30
set **interfaces** lo0 unit 0 family inet address 10.5.5.5/32

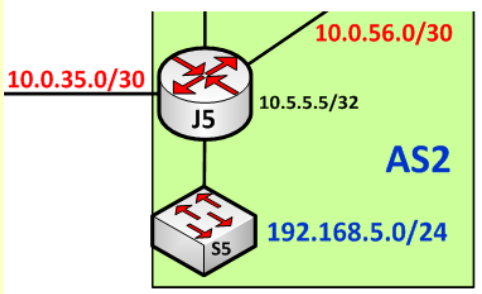
set protocols **ospf** area 0.0.0.0 interface lo0.0
set protocols **ospf** area 0.0.0.0 interface em0.0
set protocols **ospf** area 0.0.0.0 interface em5.0
set protocols **ospf** area 0.0.0.0 interface em3.0

set **routing-options** router-id 10.5.5.5
set **routing-options** autonomous-system 2

set protocols **bgp** group **PEER-AS1** type external
set protocols **bgp** group **PEER-AS1** export **To-AS1**
set protocols **bgp** group **PEER-AS1** peer-as 1
set protocols **bgp** group **PEER-AS1** neighbor 10.0.35.1

CONTINUES NEXT ...

set system host-name J5

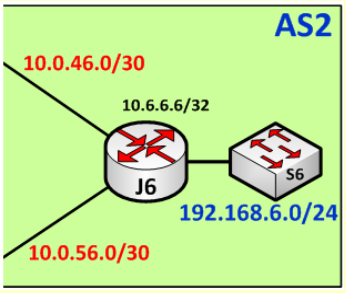


set protocols **bgp** group **INTERNAL** type internal
set protocols **bgp** group **INTERNAL** local-address 10.5.5.5
set protocols **bgp** group **INTERNAL** export **iBGP**
set protocols **bgp** group **INTERNAL** peer-as 2
set protocols **bgp** group **INTERNAL** local-as 2
set protocols **bgp** group **INTERNAL** neighbor 10.4.4.4
set protocols **bgp** group **INTERNAL** neighbor 10.6.6.6

set **policy**-options policy-statement **To-AS1** term 1 from route-filter 192.168.4.0/24 exact
set **policy**-options policy-statement **To-AS1** term 1 then accept
set **policy**-options policy-statement **To-AS1** term 2 from route-filter 192.168.6.0/24 exact
set **policy**-options policy-statement **To-AS1** term 2 then accept
set **policy**-options policy-statement **To-AS1** term 3 from route-filter 192.168.5.0/24 exact
set **policy**-options policy-statement **To-AS1** term 3 then accept

set **policy-**options policy-statement **iBGP** term 1 from protocol bgp
set **policy**-options policy-statement **iBGP** term 1 from as-path **LOCAL**
set **policy**-options policy-statement **iBGP** term 1 then reject
set **policy**-options policy-statement **iBGP** term 2 from protocol bgp
set **policy**-options policy-statement **iBGP** term 2 from as-path **EXTERIOR**
set **policy**-options policy-statement **iBGP** term 2 then next-hop self
set **policy**-options policy-statement **iBGP** term 2 then accept
set **policy**-options as-path **LOCAL "()"**
set **policy**-options as-path **EXTERIOR .***

set system host-name J6



set **interfaces** em0 unit 0 family inet address 192.168.6.1/24
set **interfaces** em3 unit 0 family inet address 10.0.56.2/30
set **interfaces** em4 unit 0 family inet address 10.0.46.2/30
set **interfaces** lo0 unit 0 family inet address 10.6.6.6/32

set **routing-options** router-id 10.6.6.6
set **routing-options** autonomous-system 2

set protocols **bgp** group **INTERNAL** type internal
set protocols **bgp** group **INTERNAL** local-address 10.6.6.6
set protocols **bgp** group **INTERNAL** local-as 2
set protocols **bgp** group **INTERNAL** neighbor 10.4.4.4 peer-as 2
set protocols **bgp** group **INTERNAL** neighbor 10.5.5.5 peer-as 2

set protocols **ospf** area 0.0.0.0 interface lo0.0
set protocols **ospf** area 0.0.0.0 interface em0.0
set protocols **ospf** area 0.0.0.0 interface em3.0
set protocols **ospf** area 0.0.0.0 interface em4.0

Chapter References

[1] BGP Best Path Selection Algorithm. **Document ID:** 13753. **Updated:** 21/5/12. www.cisco.net

[2] Configuring the Preference Value for BGP Routes. (2017).

http://www.juniper.net/techpubs/en_US/junos11.4/topics/example/bgp-preference.html

[3] Toronto Internet Exchange TorIX. (2016) https://www.torix.ca

[4] Randy Zhang. Micah Bartell. (2004). BGP Design and Implementation. Cisco Press.

CHAPTER 7

Community Attribute

Community is the most versatile and helpful BGP attribute. Essentially, the community attribute is a numeric tag, with an arranged meaning, that is attached to a route update. Based on the value of the community, routing policies take actions. Some communities have standard universal meanings but most are locally defined by network administrators. Because the local meaning can be anything, community based policies are very flexible to implement. In fact, new uses are found for communities all the time.

In a sender router, a routing policy is configured to match a set of routes that have common factors. Then a community value is attached to these routes and the routing advertisement is sent to another router. The receiving router will match the update with an inbound policy and, depending of the value of the community, it will implement an action according to the needs of the routing domain's administration.

Format of the Community Attribute

- The Community attribute **type code** is **eight (8)**.
- Community is an **Optional** attribute that it might be used or not. In fact, most communities have no effect if there is no policy to match it in the destination router. In that case, the field is just not used. A few well-known communities do have an automatic effect.
- Communities are **transitive**, so they keep propagating with the updates even if some autonomous systems in the path do nothing with the attribute. (Note: Cisco IOS requires a command to send the community).
- There are two **formats** for Community: **standard** and **extended** and they have different lengths.

Standard Community

- The **field length** of the Standard Community Attribute is 32 bits or **4 bytes**.
- The **format** of the attribute is **2 bytes: 2 bytes**.
- The **typical use** of the attribute is that the first two bytes is set to the autonomous system number and the other two bytes is the specific value of the attribute. For example, for ASN 65,144, community 10, it would be 65144:10. This is a common practice, but the two fields could actually be any number (excluding special cases).
- Special cases. The range that begins with the first two bytes set to 0 is reserved:
    - From 0000000_{Hex} thru $0000FFFF_{Hex}$ (Decimal 0 thru 65,535).
- The range that begins with the first two bytes set to 65,535 is reserved:
    - From $FFFF0000_{Hex}$ thru $FFFFFFFF_{Hex}$ (Decimal 4,294,901,760 thru 4,294,967,295).

Community Attribute Case Topology

The study of the community attribute starts with the simple topology shown next that includes the configuration for communities in both Cisco and Juniper routers. The left half of the network has Cisco routers and the right half has Juniper routers. In Cisco R1, communities are set for the two routes 192.168.1.0/24 and 192.168.2.0/24. Similarly, communities are set in Juniper J4 for 172.16.3.0/24 and 172.16.4.0/24. The autonomous system AS2 acts just as a transit AS. The configurations of each router will be explained one by one starting from R1.

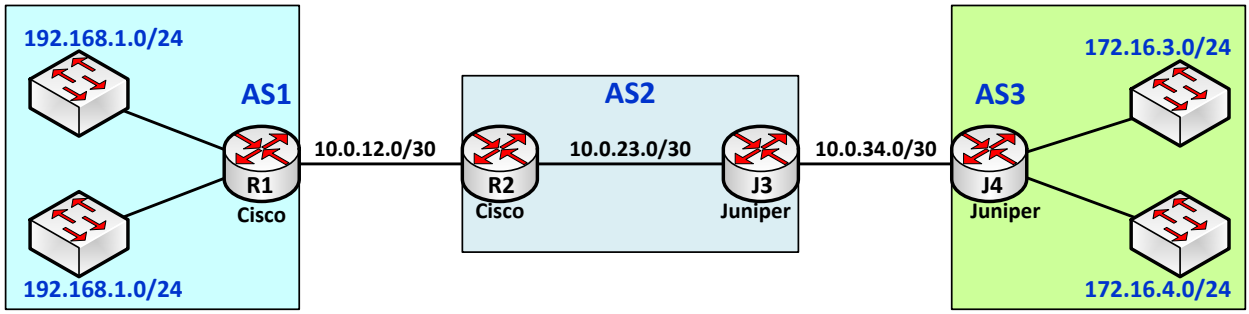
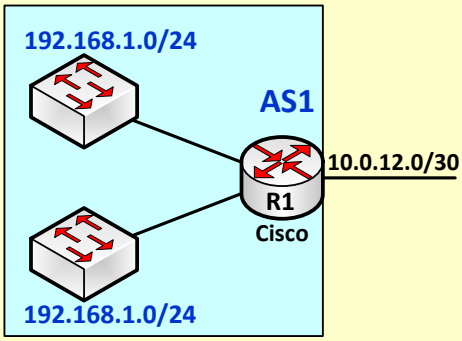


Figure 7.1

Let's begin with the configuration of Cisco router R1. Only the relevant sections of the configuration are presented here. The snippets contain the commands and its explanation.




```
router bgp 1
 network 192.168.1.0
 network 192.168.2.0
 neighbor 10.0.12.2 remote-as 2
 neighbor 10.0.12.2 send-community
 neighbor 10.0.12.2 route-map To-AS2 out
!
route-map To-AS2 permit 10
 match ip address 1
 set community 1:1
!
route-map To-AS2 permit 20
 match ip address 2
 set community 1:2
!
ip bgp-community new-format
!
access-list 1 permit 192.168.1.0 0.0.0.255
access-list 2 permit 192.168.2.0 0.0.0.255
```

R1 parses its routing updates toward R2, thru the policy **To-AS2**.

The command "send-community" must be set, otherwise the community value is not delivered.

The route-map **To-AS2** selects the routes and attaches the respective communities that will be carried by the BGP UPDATE. The route 192.168.1.0/24 will have the Community 1:1 and the route 192.168.2.0/24 will have the Community 1:2

The command "new-format" fixes the presentation of the community value in Cisco IOS. Otherwise, the community 1:1 will show as 65537 and the community 1:2 will show as 65538. It works also but it is just harder to read.

Figure 7.2

The result of this configuration can be observed in the sniffed BGP UPDATE sniffed in R1's outbound interface. The route being advertised is 192.168.1.0/24 and beside the regular attributes, the UPDATE carries the community attribute 1:1. The basic parameters of the attribute, optional, transitive, complete, type, and length are described. Notice that there will be a different UPDATE for the other prefix 192.168.2.0/24 since it has a different attribute 1:2.

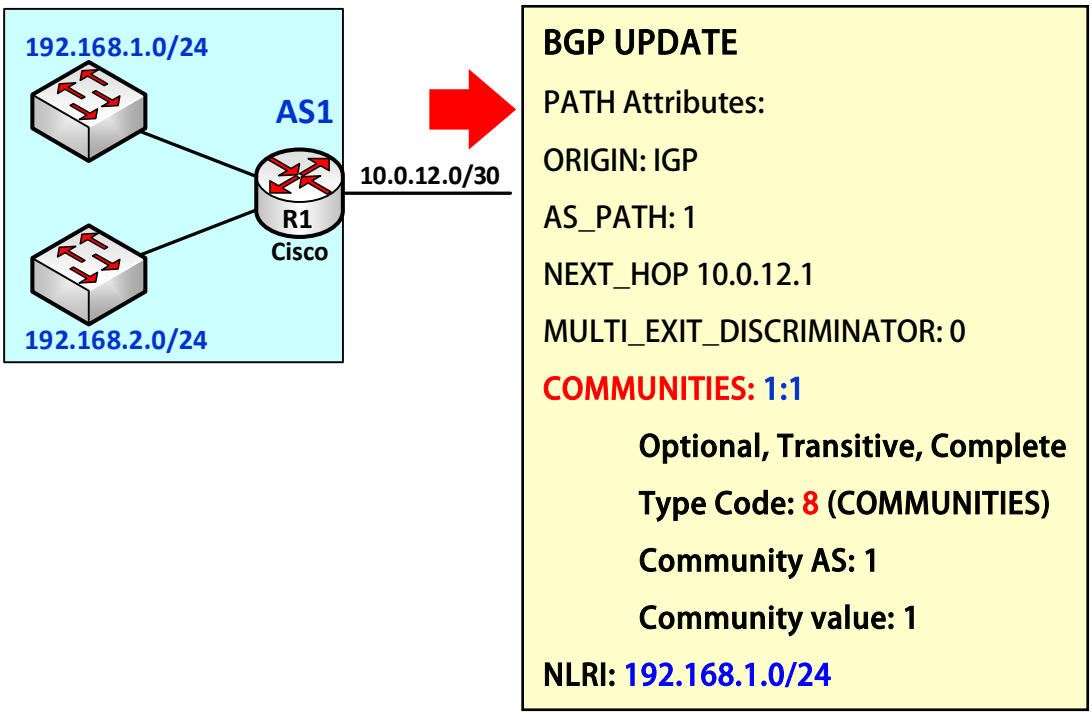
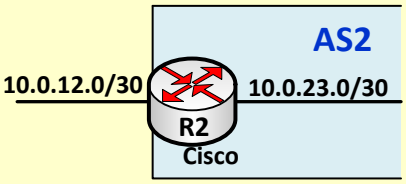


Figure 7.3

Currently, no policies will be enacted in router R2, so it is just left with a minimum configuration. Even if this router is not doing anything with the community, it is configured to let it go thru. This is done with the command "send-community". Cisco IOS has this mode of operation while Juniper just allows the community to be propagated with the BGP updates (as it should, being the attribute transitive).

hostname R2



router bgp 2

neighbor 10.0.12.1 remote-as 1

neighbor 10.0.12.1 **send-community**

neighbor 10.0.23.2 remote-as 2

neighbor 10.0.23.2 next-hop-self

neighbor 10.0.23.2 **send-community**

!

ip bgp-community new-format

!

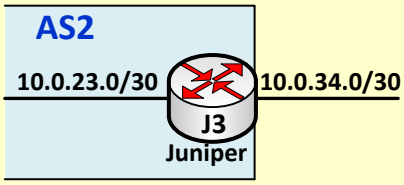
Initially, R2 has a simple configuration to just allow the routing UPDATE to pass thru.

The command "send-community" is being set to all the neighbours so the community values can propagate in all directions.

Figure 7.4

The following snippets show the BGP configuration of Juniper J3. This router is just allowing the BGP routing information to propagate. No action is being taken upon the community attribute at this time.

hostname R2




```
set routing-options router-id 10.3.3.3
set routing-options autonomous-system 2
edit protocols bgp group to-AS3
        set type external
        set neighbor 10.0.34.2 peer-as 3
        top
edit protocols bgp group AS2
        set type internal
        set neighbor 10.0.23.1 peer-as 2
        set export INTERNAL
        top
edit policy-options policy-statement INTERNAL
        set term 1 from protocol bgp
        set term 1 then next-hop self
        set term 1 then accept
        set term 2 then reject
```

Juniper router J3 has two BGP groups defined, one to peer with router J4 in AS3 and the other for internal BGP peering with R2.

The only policy configured, called INTERNAL, is for iBGP reachability to set the next-hop to self.

No community configuration is needed at this point.

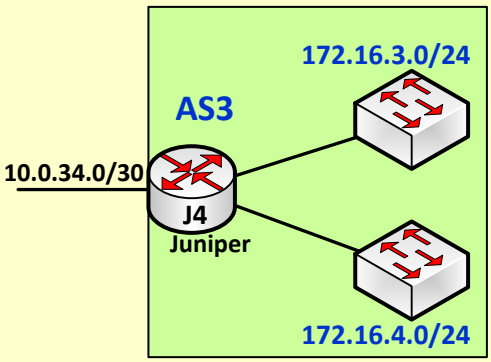
Notice that JUNOS routers allow the community attribute to propagate by default.

Figure 7.5

Finally, J4 configuration shows how to configure community values in Juniper. This router exports the prefix 172.16.3.0/24 with the community 3:1 and the prefix 172.16.4.0/24 with the

community 3:2. The technique that Junos uses is to match the prefixes with one of the route filtering tools, in this case a route-filter. Then the corresponding community string is added. However, the string has to be defined in the policy-option to match the numerical value of the attribute.

hostname J4




```
set routing-options router-id 10.4.4.4
set routing-options autonomous-system 3
edit protocols bgp group to-AS2
        set type external
        set neighbor 10.0.34.1 peer-as 2
        set export to-AS2
        top
edit policy-options policy-statement to-AS2
        set term 1 from route-filter 172.16.3.0/24 exact community-add COMM1
        set term 1 then accept
        set term 2 from route-filter 172.16.4.0/24 exact community-add COMM2
        set term 2 then accept
        top
set policy-options community COMM1 members 3:1
set policy-options community COMM2 members 3:2
```

Figure 7.6

Since all the routers have been configured, it is time to check the results. Router R1 receives the information coming from router J4 via BGP updates. These two prefixes, 172.16.3.0/24 and 172.16.4.0/24 have their community values attached.

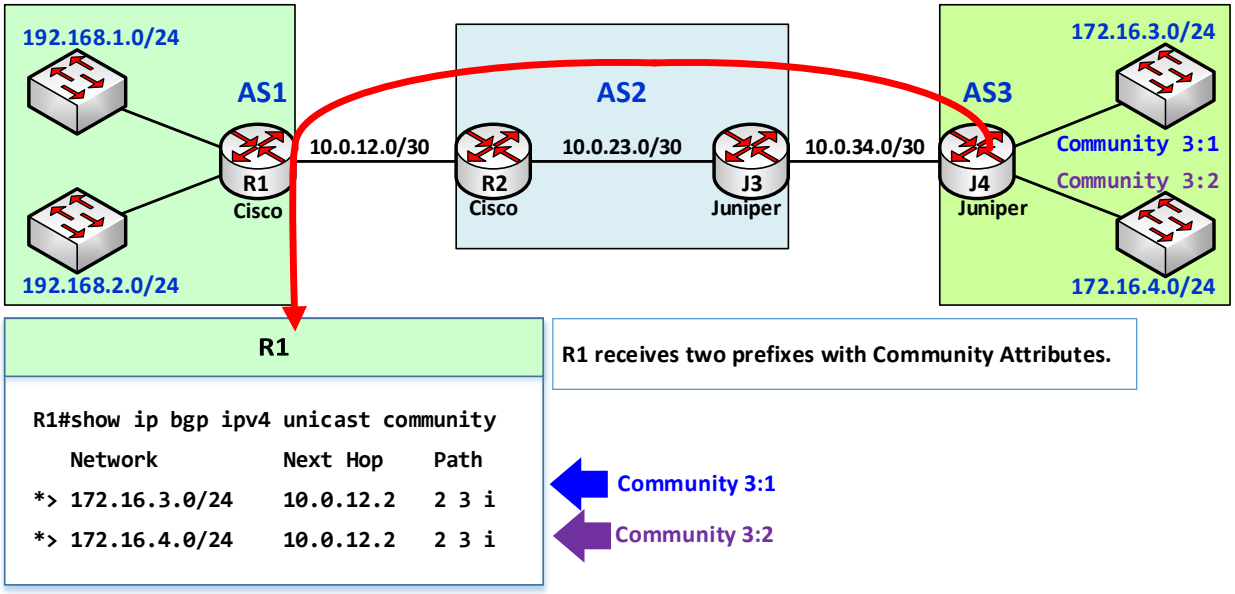


Figure 7.7

Router R1 is not implementing any action based on the attributes just yet. This is only a demonstration of how Communities are set and propagated. This also demonstrates the **transitive** property of the attribute since it went thru AS2 even though AS2 is doing nothing with the attribute.

On the other side, Juniper router J4 knows the prefixes advertised by R1 with the communities 1:1 and 1:2.

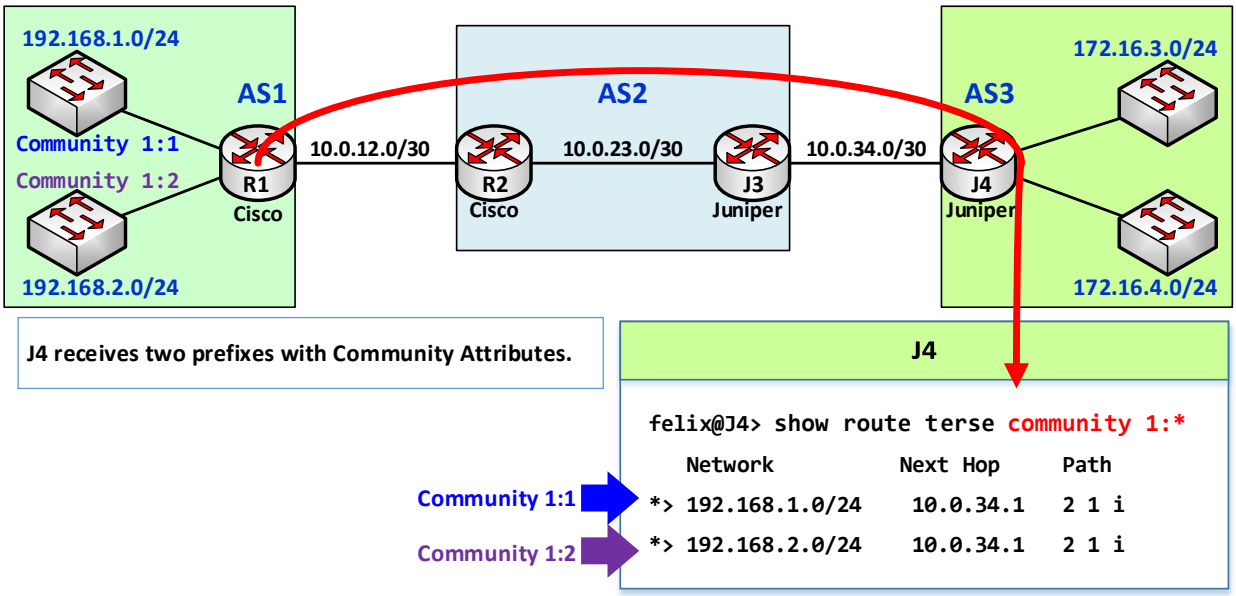
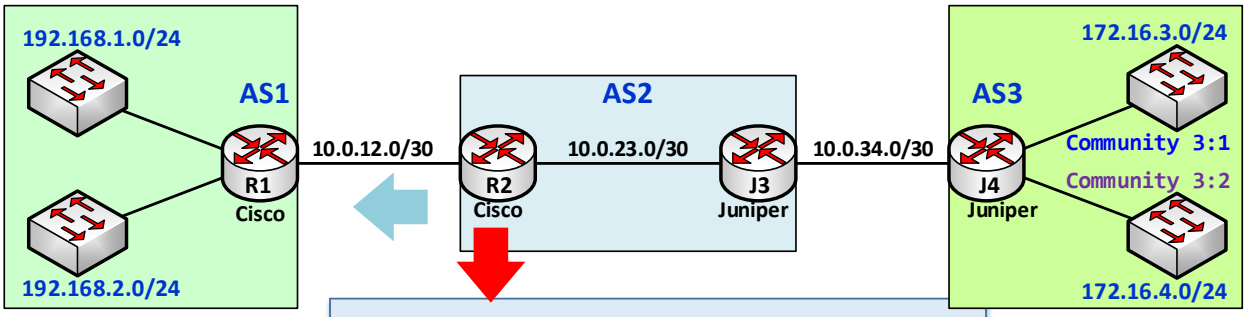


Figure 7.8

Implementing Policies with Communities

A **locally defined community** has no effect if there is no policy to match and make a routing decision. In the previous sections was demonstrated how the community values were assigned and then carried between AS1 and AS3 thru AS2. However, they were not used for any particular application.

In this section, let's configure policies to make decisions based on the values of the communities. Since routers R1 and J4 have already attached communities to certain routes, the border routers in AS2 will be configured to match these communities and to set metrics. Cisco router R2 will apply a policy outbound toward AS1 matching the communities sourced by AS3. For community 3:1 attached to route 172.16.3.0/24, router R2 sets a metric of 3 and for community 3:2 attached to route 172.16.4.0/24, the metric will be 4.

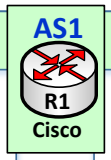


R2 outbound policy to R1 in AS1

```
router bgp 2
 neighbor 10.0.12.1 remote-as 1
 neighbor 10.0.12.2 send-community
 neighbor 10.0.12.2 route-map EXPORT-to-AS1 out
 neighbor 10.0.23.2 remote-as 2
 neighbor 10.0.23.2 next-hop-self
 neighbor 10.0.23.2 send-community
!
route-map EXPORT-to-AS1 permit 10
 match community 1
 set metric 3
!
route-map EXPORT-to-AS1 permit 20
 match community 2
 set metric 2
!
ip community-list 1 permit 3:1
ip community-list 2 permit 3:2
!
ip bgp-community new-format
```

Figure 7.9

The result of the policy can be observed in router R1:



R1#show ip bgp (BEFORE)

BGP table version is 12, local router ID is 10.1.1.1

Network	Next Hop	Metric	Path
*> 10.4.4.4/32	10.0.12.2		2 3 i
***> 172.16.3.0/24**	**10.0.12.2**		**2 3 i**
***> 172.16.4.0/24**	**10.0.12.2**		**2 3 i**
*> 192.168.1.0	0.0.0.0	0	
*> 192.168.2.0	0.0.0.0	0	

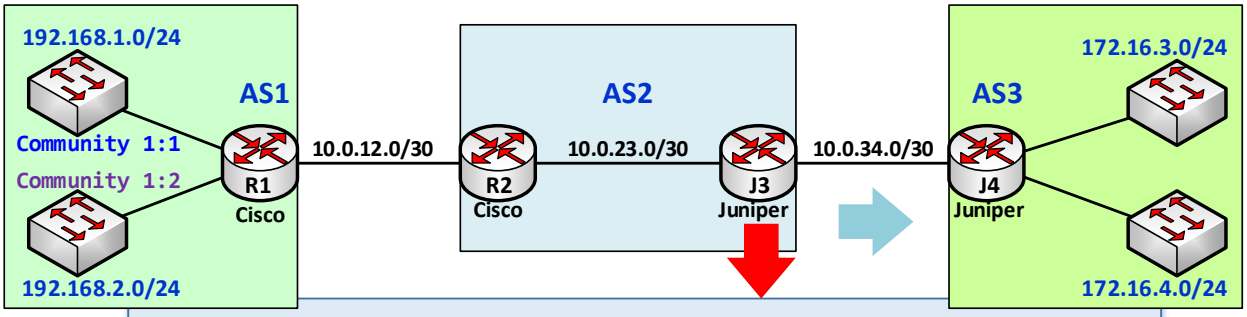
R1#show ip bgp (AFTER)

BGP table version is 13, local router ID is 10.1.1.1

Network	Next Hop	Metric	Path
*> 10.4.4.4/32	10.0.12.2		2 3 i
***> 172.16.3.0/24**	**10.0.12.2**	**3**	**2 3 i**
***> 172.16.4.0/24**	**10.0.12.2**	**4**	**2 3 i**
*> 192.168.1.0	0.0.0.0	0	
*> 192.168.2.0	0.0.0.0	0	

Figure 7.10

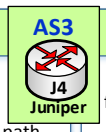
A mirror configuration is implemented in router J3 to demonstrate the Juniper way. In effect, J3 matches the communities advertised by R1-AS1 and then it sets metrics. Specifically, J3 matches community 1:1 attached to route 192.168.1.0/24 to set a metric of 11. Also, J3 matches community 1:2 attached to route 192.168.2.0/24 to set a metric of 12. The policy is applied outbound toward J4 where the result is observed. The following diagrams shows the procedure and result.



J3 outbound policy toward J4 in AS3

```
set protocols bgp group to-AS3 type external
set protocols bgp group to-AS3 neighbor 10.0.34.2 peer-as 3
set protocols bgp group to-AS3 export MATCH_COMM

set policy-options policy-statement MATCH_COMM term 1 from community AS1-SW1
set policy-options policy-statement MATCH_COMM term 1 then metric 11
set policy-options policy-statement MATCH_COMM term 1 then accept
set policy-options policy-statement MATCH_COMM term 2 from community AS1-SW2
set policy-options policy-statement MATCH_COMM term 2 then metric 12
set policy-options policy-statement MATCH_COMM term 2 then accept

set policy-options community AS1-SW1 members 1:1
set policy-options community AS1-SW1 members 1:2
```

Figure 7.11



R1#show ip bgp (BEFORE)

felix@J4> show route terse

A Destination	P	Prf	Metric 1	Metric 2	Next hop	AS path
* 10.4.4.4/32	D	0			>lo0.0	
* 10.0.34.0/30	D	0			>em3.0	
* 172.16.3.0/24	D	0			>em0.0	
* 172.16.4.0/24	D	0			>em1.0	
*** 192.168.1.0/24**	**B**	**170**	**100**		**>10.0.34.1**	**2 1 I**
*** 192.168.2.0/24**	**B**	**170**	**100**		**>10.0.34.1**	**2 1 I**

R1#show ip bgp (AFTER)

felix@J4> show route terse

A Destination	P	Prf	Metric 1	Metric 2	Next hop	AS path
* 10.4.4.4/32	D	0			>lo0.0	
* 10.0.34.0/30	D	0			>em3.0	
* 172.16.3.0/24	D	0			>em0.0	
* 172.16.4.0/24	D	0			>em1.0	
*** 192.168.1.0/24**	**B**	**170**	**100**	**11**	**>10.0.34.1**	**2 1 I**
*** 192.168.2.0/24**	**B**	**170**	**100**	**12**	**>10.0.34.1**	**2 1 I**

Figure 7.12

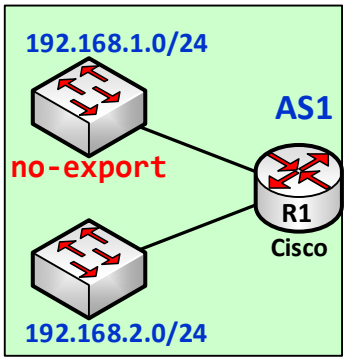
Router J4 shows the IP route 192.168.1.0/24 with a metric of 11 and the IP route 192.168.2.0/24 with a metric of 12 just as the policy intended. So this demonstrate the technique to set an action based on a community. The topology being used here is too simple to show an effect, so later on, a more complex example will demonstrate the benefits of communities.

Well-Known Communities

These are several communities with pre-established values and meanings defined in RFC documents. Consequently, all the vendors must implement the understanding of these communities in their software. The advantage of these communities is that the routers automatically respond to them without the need of policies. The previous network will be modified to show the functioning of two of these community values. The IP prefix 192.168.1.0/24 will be marked with NO_EXPORT and the IP prefix 192.168.2.0/24 will be marked with NO_ADVERTISE.

Do Not Export

The NO_EXPORT community has a value of $FFFFFF01_{hex}$ or $4,294,967,041_{dec}$ but all vendors have a command to activate it so there is no need to remember the value. This well-known community means "**do not export to the next Autonomous System**." If a router receives this community, it can propagate the route to other routers inside the same AS, but the border routers will not further propagate the route. The following diagram shows how R1 marks the IP prefix 192.1681.1.0/24 with the community "NO_EXPORT".




```
hostname R1
route-map EXPORT-to-AS2 permit 10
 match ip address 1
 set community no-export
!
access-list 1 permit 192.168.1.0 0.0.0.255
```

Figure 7.13

Router R2 receives the update, installs the route and then it proceeds to relay the information to the AS border router J3 as shown in sniffed BGP UPDATE messages.

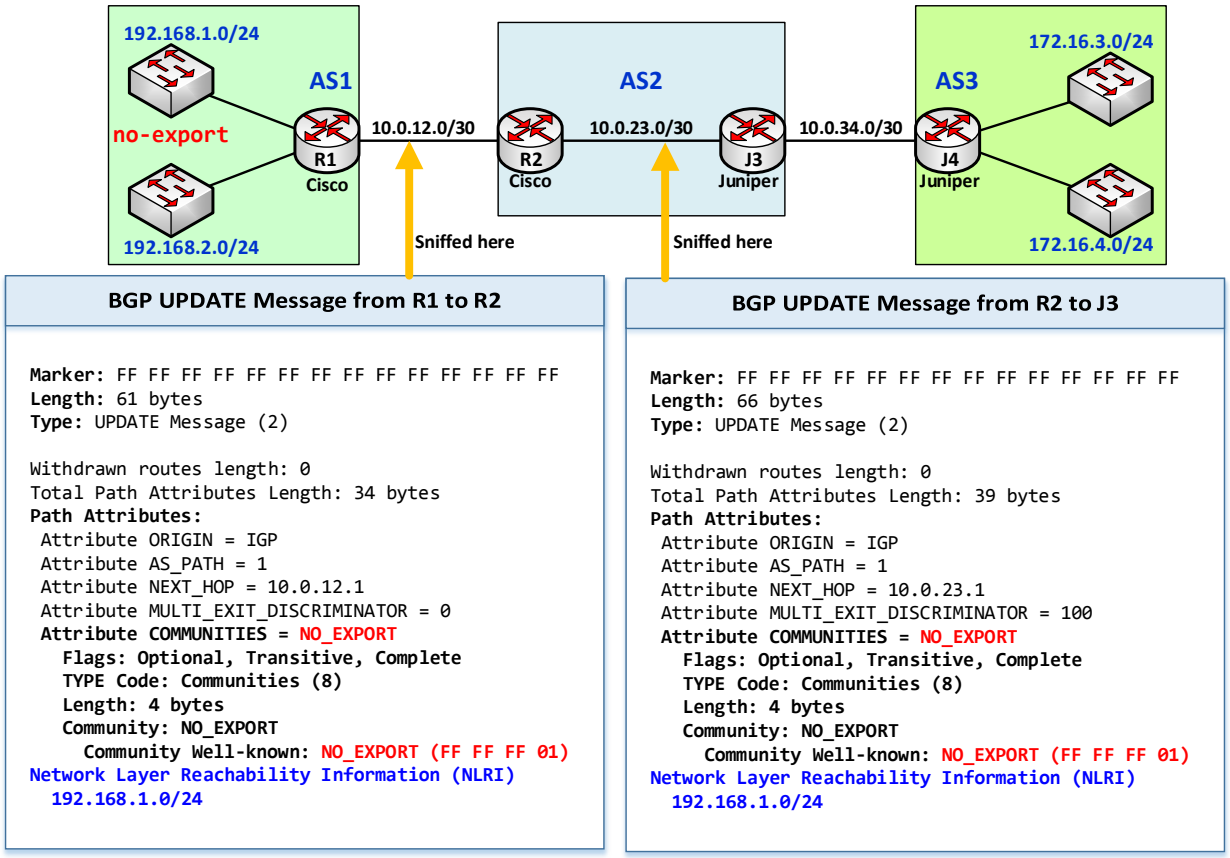


Figure 7.14

J3 knows192.168.1.0/24 with the community no-export but it will not export such prefix to J4 in AS3 because that is precisely the point of this attribute.

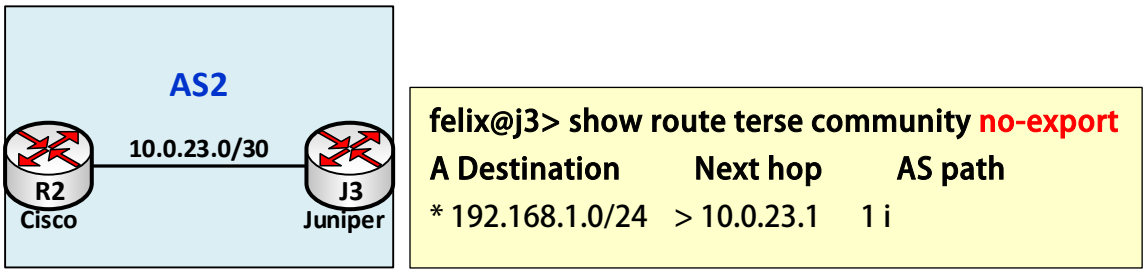


Figure 7.15

Do Not Advertise

The NO_ADVERTISE community has a value of $FFFFFF02_{hex}$ or $4,294,967,042_{dec}$ and all vendors have a command to activate it. This well-known community means "do not advertise to any peer". If a router receives an update with a prefix marked with this community, it will keep the information to itself. The following diagram contains plenty of information about the functioning of the NO_ADVERTISE attribute.

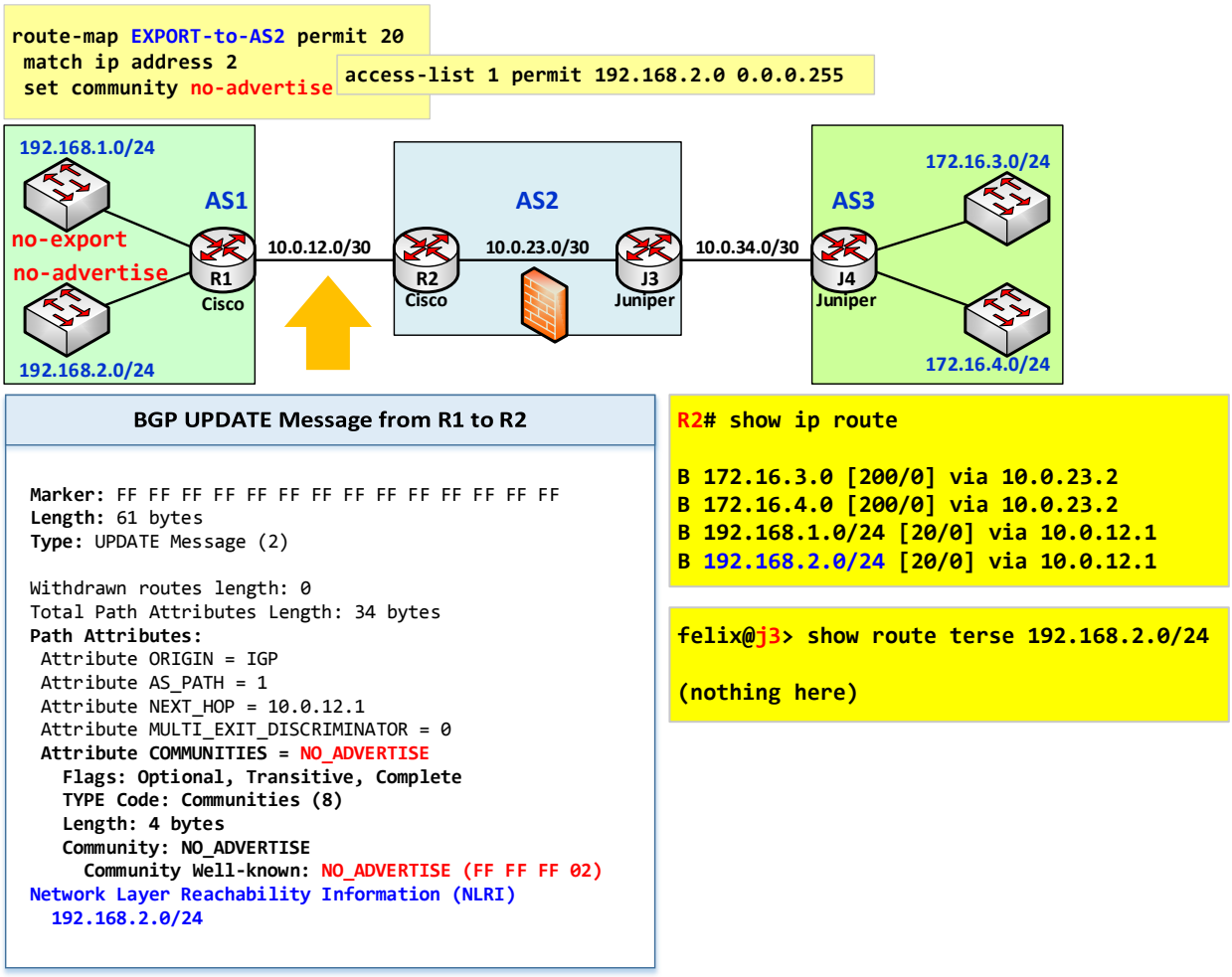


Figure 7.14

- Router R1 matches the prefix 192.168.2.0/24 and sets the Well Known Community "DO NOT ADVERTISE", then it sends the routing UPDATE that it is shown to the left.
- Router R2 receives the UPDATE and installs the IP route 192.168.2.0/24 in its routing table; However, R2 does not re-advertise the route to the internal neighbour J3 as the snippet above proves it.

Manipulating traffic paths with Communities

An (ISP) Internet Service Provider AS2 gets upstream access to the rest of Internet via a larger carrier AS4. In turn, AS2 offers IP connectivity to its customers, in this case AS1 and AS3. The ISP has several services profiles shown in the next diagrams. The service to the left side offers to interconnect certain networks between AS1 and AS3 without being advertised to the upstream path. On the right, certain customer networks gain access to the global Internet.

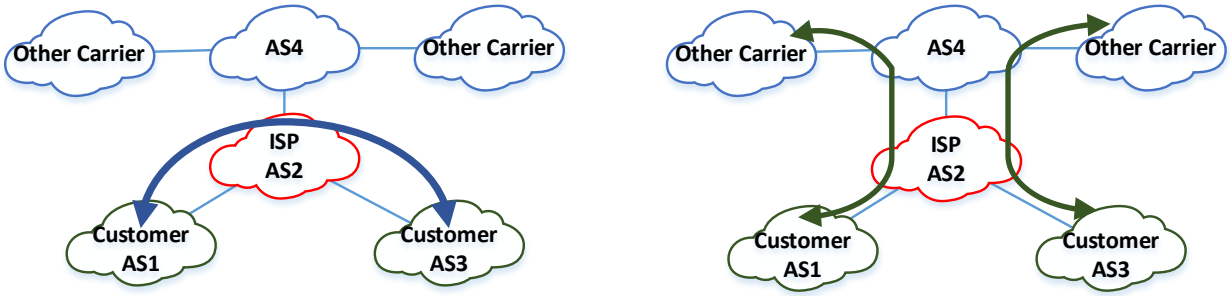


Figure 7.15

The previous generic scenario can be implemented as it follows

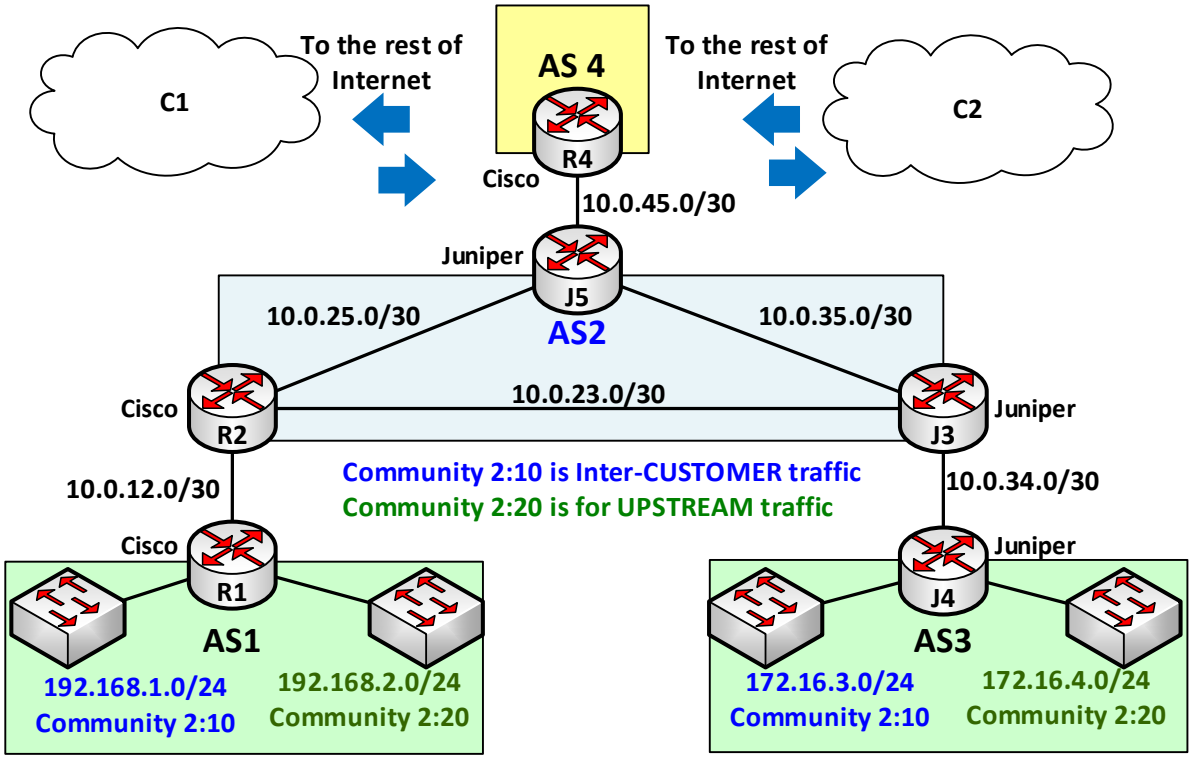
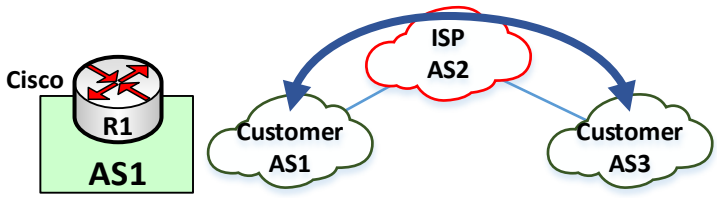


Figure 7.16

In this example, AS1 and AS2 want traffic to flow between the networks 192.168.1.0/24 and 172.16.3.0/24. However, imagine that the customers eventually want to add more networks to this shared traffic interest without asking the ISP for reconfiguration. Furthermore, the ISP (AS2) does not want to be reconfiguring every time that a customer adds a network. Consequently, the ISP sets a service policy that states "if you want to exchange traffic with other customers like you, just set this community 2:10 and AS2 will match it". Hence, a community is set apart for common traffic interest between customers only.

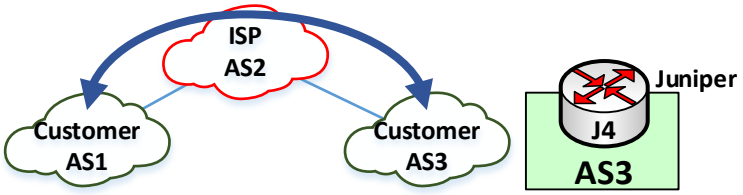
First, the Inter-customer traffic policy configuration is shown next. The customer border routers R1-AS1 and J4-AS3 mark the routes 192.168.1.0/24 and 172.16.3.0/24 with the community value 2:10.




```
router bgp 1
 network 192.168.1.0
 network 192.168.2.0
 neighbor 10.0.12.2 remote-as 2
 neighbor 10.0.12.2 send-community
 neighbor 10.0.12.2 route-map EXPORT-to-AS2 out
!
access-list 1 permit 192.168.1.0 0.0.0.255
access-list 2 permit 192.168.2.0 0.0.0.255
!
route-map EXPORT-to-AS2 permit 10
 description "Set community 2:10 for Inter-Customer traffic"
 match ip address 1
 set community 2:10
!
route-map EXPORT-to-AS2 permit 20
 description "Set community 2:20 for Upstream traffic"
 match ip address 2
 set community 2:20
```

Figure 7.17

Now, the versatility of communities becomes more evident. Communities represent groups that share common factors so the ISP configures and applies one policy to all its customers and a different policy toward the upstream path. The need for reconfiguration is reduced because any routing information carrying a community will be treated the same by the common policy. Let's explain now the implementation of the two policies.




```
set protocols bgp group to-AS2 type external
set protocols bgp group to-AS2 export EXPORT-to-AS2
set protocols bgp group to-AS2 neighbor 10.34.34.1 peer-as 2

edit policy-options policy-statement EXPORT-to-AS2
  set term 1 from route-filter 172.16.3.0/24 exact community add Inter-CUST
  set term 1 then accept
  set term 2 from route-filter 172.16.4.0/24 exact community add UPSTREAM
  set term 2 then accept

set policy-options community Inter-CUST members 2:10
set policy-options community UPSTREAM members 2:20
```

Figure 7.18

The service provider AS2 receives the BGP UPDATES, matches the community values, and filters the route out according to the predefined traffic interest. (a metric has been added just to observe it in the resulting routing tables although the metric does not achieve anything in this topology).

Cisco Router R2

```
router bgp 2
 neighbor 10.12.12.1 remote-as 1
 neighbor 10.12.12.1 send-community
 neighbor 10.12.12.1 route-map CUSTOMER out
 neighbor 10.23.23.2 remote-as 2
 neighbor 10.23.23.2 next-hop-self
 neighbor 10.23.23.2 send-community
 neighbor 10.25.25.1 remote-as 2
 neighbor 10.25.25.1 next-hop-self
 neighbor 10.25.25.1 send-community
!
ip bgp-community new-format
ip community-list 1 permit 2:10
ip community-list 2 permit 2:20
!
route-map CUSTOMER permit 10
 match community 1
 set metric 1
```

Juniper Router J3

```
set protocols bgp group INTERNAL type internal
set protocols bgp group INTERNAL export INTERNAL
set protocols bgp group INTERNAL neighbor 10.23.23.1 peer-as 2
set protocols bgp group INTERNAL neighbor 10.35.35.1 peer-as 2

set protocols bgp group CUSTOMER type external
set protocols bgp group CUSTOMER export TO-CUSTOMER
set protocols bgp group CUSTOMER neighbor 10.34.34.2 peer-as 3

set policy-options policy-statement INTERNAL term 1 from protocol bgp
set policy-options policy-statement INTERNAL term 1 then next-hop self
set policy-options policy-statement INTERNAL term 1 then accept
set policy-options policy-statement INTERNAL term 2 then reject

set policy-options policy-statement TO-CUSTOMER term 1 from community Inter-CUST
set policy-options policy-statement TO-CUSTOMER term 1 then metric 1
set policy-options policy-statement TO-CUSTOMER term 1 then accept
set policy-options policy-statement TO-CUSTOMER term 2 then reject

set policy-options community Inter-CUST members 2:10
set policy-options community UPSTREAM members 2:20
```

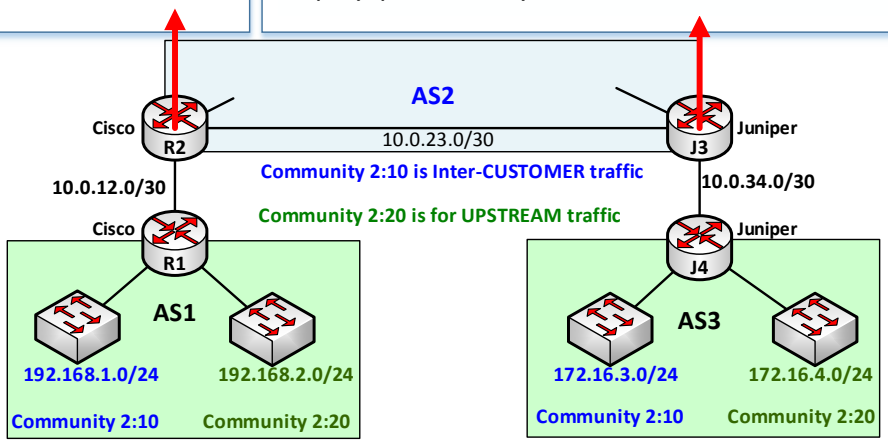


Figure 7.19

The result of the policy can be observed in the BGP databases of the customer border routers, R1 in AS1 and J4 in AS3. Observe that so far, the only IP routing information presented via BGP is the one that was marked with the communities and that is exactly what was wanted.

Router R1 in AS1

```
R1#show ip route bgp 1

B      172.16.3.0 [20/1] via 10.12.12.2
```

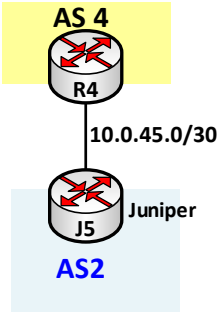
Router J4 in AS3

```
felix@J4> show route protocol bgp terse

A Destination        P Prf   Metric 1   Metric 2   Next hop     AS path
* 192.168.1.0/24     B 170     100          1      >10.34.34.1  2 1 I
```

Figure 7.20

The next step is to configure and test another policy which will allow the routes 192.168.2.0/24 in AS1 and 172.16.4.0/24 in AS3 to be advertised to the upstream AS4. In consequence, a policy is implemented in router J5 to advertise to the upstream R4 in AS4 only the routes marked with the community 2:20.

Juniper Router J5




```
set routing-options router-id 10.5.5.5
set routing-options autonomous-system 2

set protocols bgp group INTERNAL type internal
set protocols bgp group INTERNAL export INTERNAL
set protocols bgp group INTERNAL neighbor 10.0.25.2 peer-as 2
set protocols bgp group INTERNAL neighbor 10.0.35.2 peer-as 2

set policy-options policy-statement INTERNAL term 1 from protocol bgp
set policy-options policy-statement INTERNAL term 1 then next-hop self
set policy-options policy-statement INTERNAL term 1 then accept
set policy-options policy-statement INTERNAL term 2 then reject

set protocols bgp group UPSTREAM type external
set protocols bgp group UPSTREAM export UPSTREAM
set protocols bgp group UPSTREAM neighbor 10.0.45.1 peer-as 4

set policy-options policy-statement UPSTREAM term 1 from protocol bgp
set policy-options policy-statement UPSTREAM term 1 from community TO-
UPSTREAM
set policy-options policy-statement UPSTREAM term 1 then metric 2
set policy-options policy-statement UPSTREAM term 1 then accept
set policy-options policy-statement UPSTREAM term 2 then reject

set policy-options community TO-UPSTREAM members 2:20
```

Figure 7.21

The result of the policy is verified in router R4.

Router R4 in AS4

```
R4#show ip bgp
local router ID is 10.4.4.4

   Network          Next Hop    Metric LocPrf Weight  Path
*> 172.16.4.0/24    10.0.45.2     2             0     2 3 i
*> 192.168.2.0      10.0.45.2     2             0     2 1 i
```

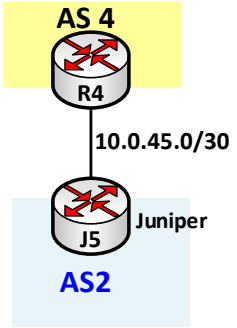


Figure 7.22

This example has illustrated that routing information can be better controlled when the decisions are made based on the community that identifies a group of routes instead of configuring policies based only in filtering IP addresses and AS numbers.

Chapter Summary

The community attribute is the most appreciated BGP capability by network engineers when they have to implement policies. In fact, communities are so important that all large Internet organizations have well-developed plans of usage of communities which are typically made public. Communities enable routing polices by matching common factors instead of only matching IP prefixes and AS paths.

The most common communities follow the Standard format and they are structured in two hexadecimal strings of 2 bytes each separated by a colon. Users of communities typically set the first byte to the value of the autonomous system and then the second byte to an arbitrary number. The community string is then assigned a meaning bound to a routing intention within an organization.

There are some reserved values and special meaning communities described in the Request for Comments. Some of these communities have universal meaning.

Exercises

7.1 Configure the following network topology:

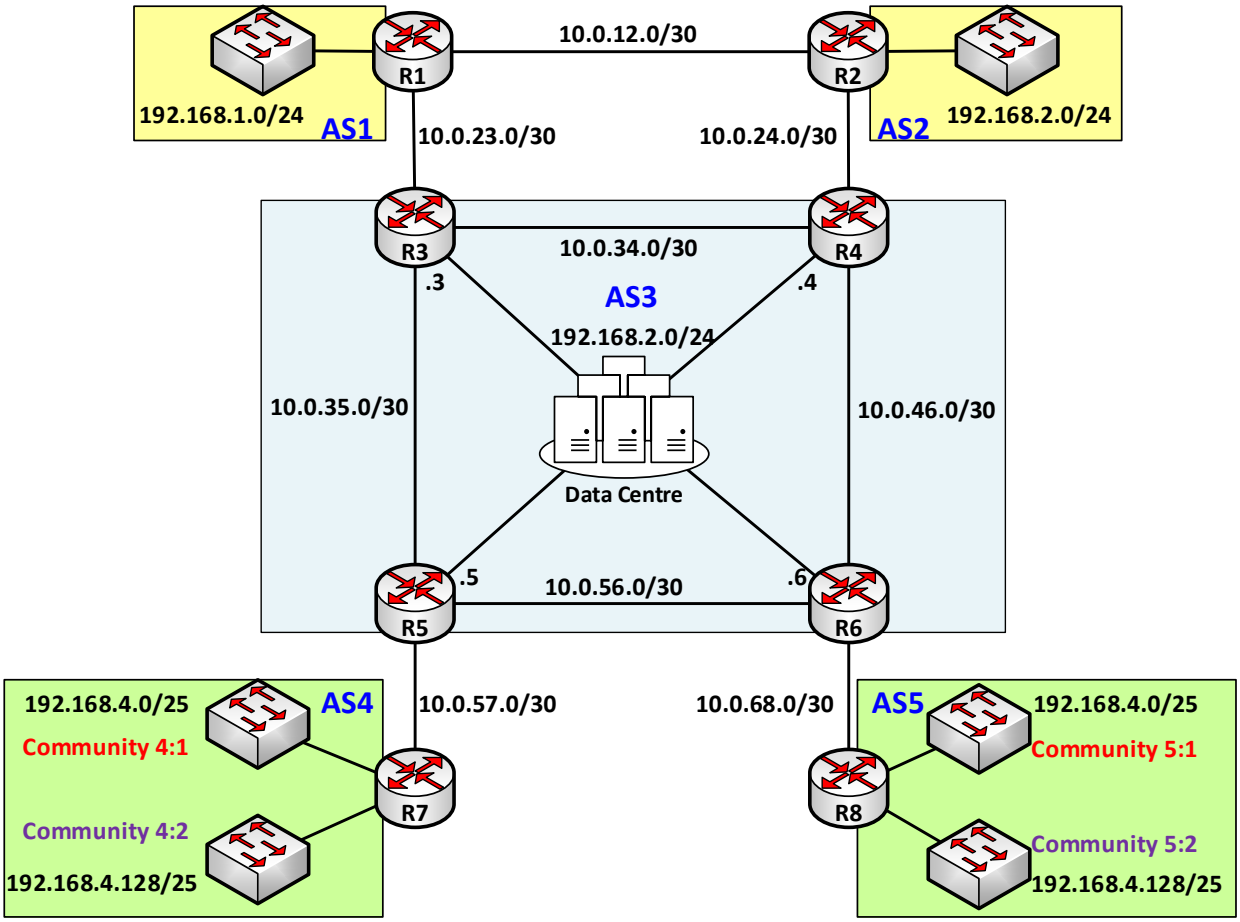


Figure 7.23

- The networks within communities 4:1 and 5:1 must reach the data centre in AS3 only.
- The networks with communities 5:1 and 5:2 must reach each other but AS1 and AS2 must not learn about these networks.
- In case that the link 10.0.12.0/30 failed, the networks in AS1 and AS2 must still be reachable by following any possible path within AS3. However, AS4 and AS5 should not learn about this networks. Implement this with communities.

7.2 Configure the following network topology:

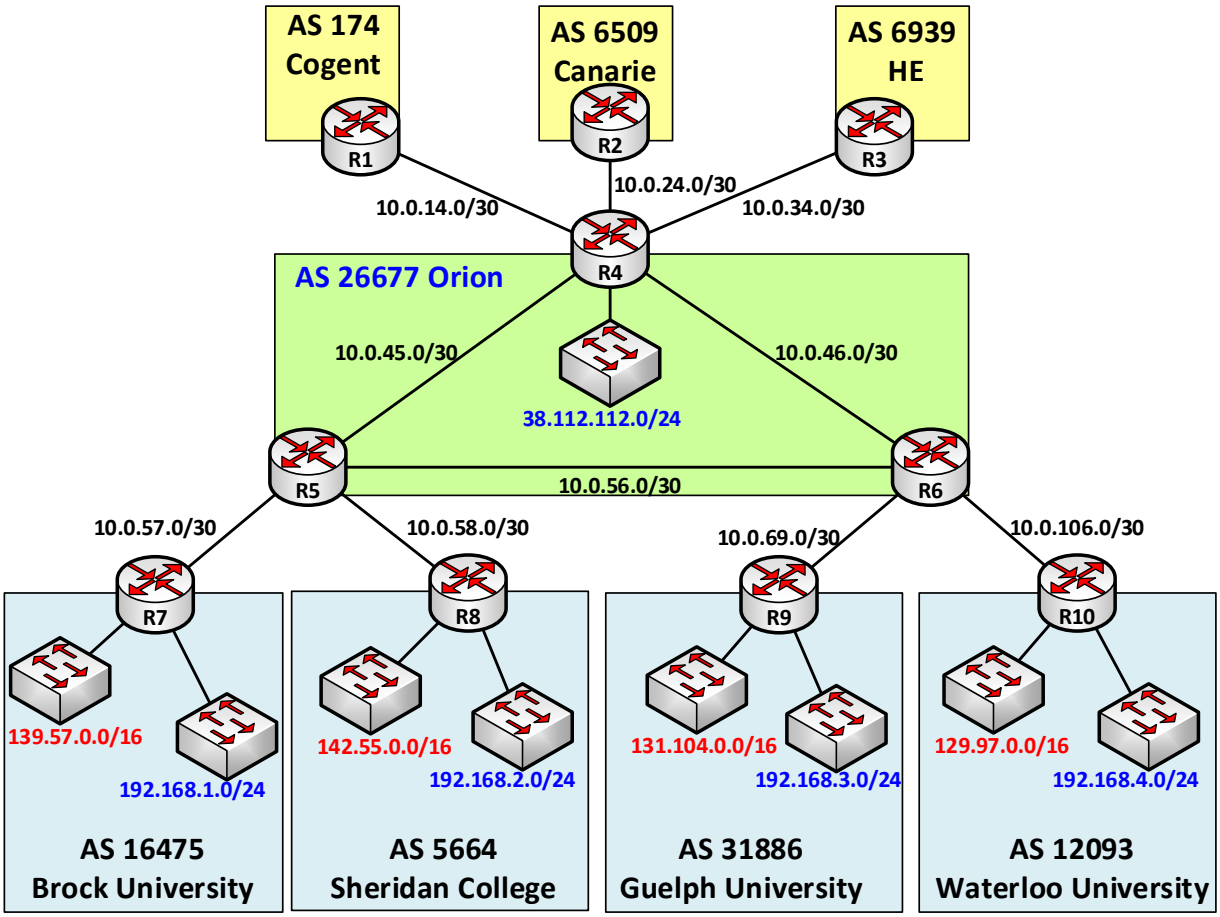


Figure 7.24

- Use locally defined communities and well-known communities such as Internet, Do-not-Export, and Do-not-advertise to implement the following policies:
- Brock University, Sheridan College, Guelph University, and Waterloo University would want to establish a cooperation network using the networks 192.168.1.0/24, 192.168.2.0/24, 192.168.3.0/24, 192.168.4.0/24.
- Orion AS 26677 agrees to advertise such prefixes as long as the 4 members of the community advertise them with the same specific community.
- However, Orion must not advertise such prefixes to Cogent, Canarie and Hurricane Electric HE).
- Orion AS 26677 should know all the other prefixes (139.57.0.0/16, 142.55.0.0/16, 131.104.0.0/16, and 129.97.0.0/16).
- Cogent and HE should only learn 38.112.112.0/24 which is sourced in Orion.

7.3) Implement the following topology by following the routing intentions described below:

Disclaimer: this is a made-up network for educational purposes only. Although some of the AS numbers and company names are used for more realism.

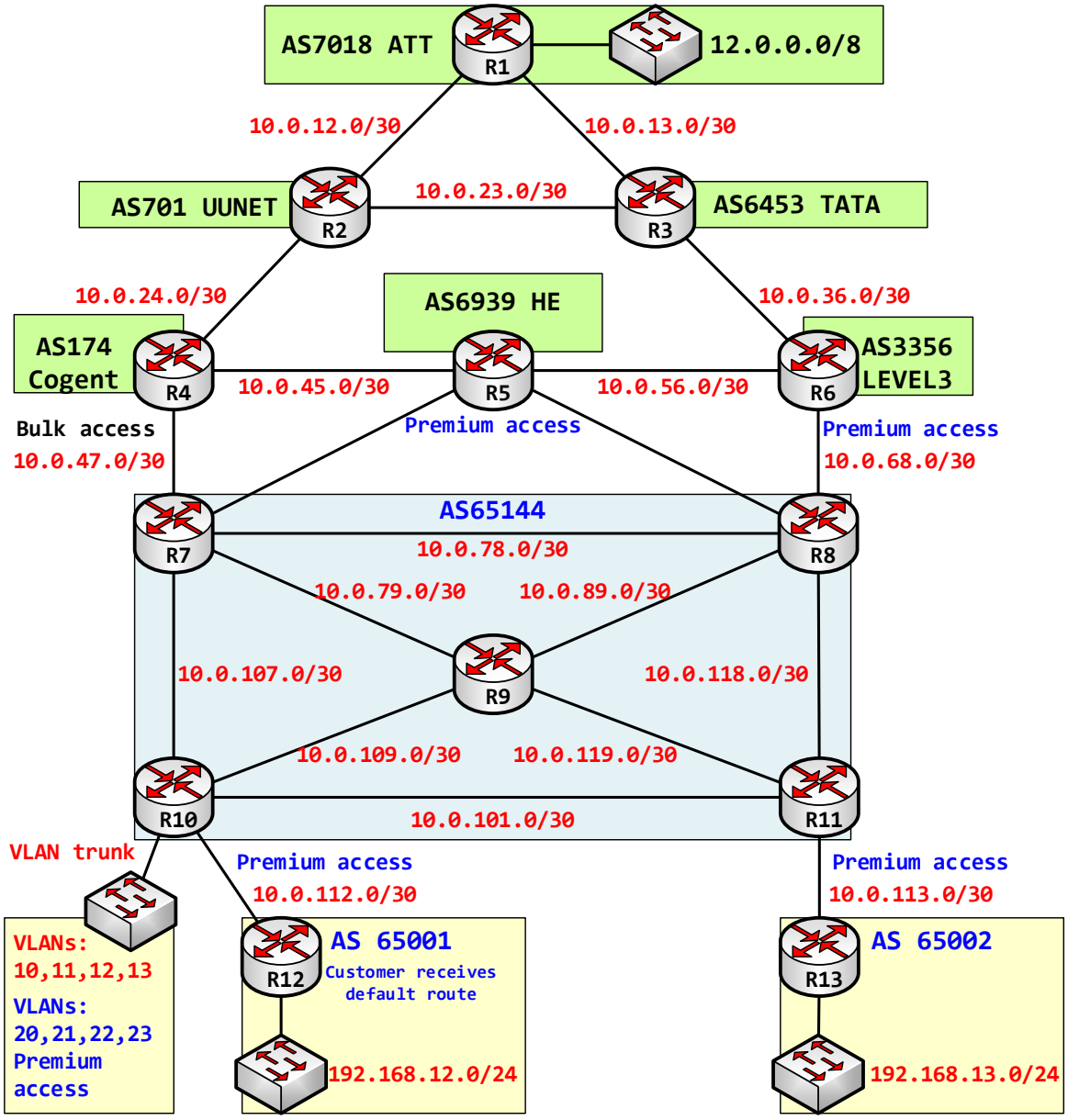


Figure 7.25

- The fictitious AS 65144 would be an Internet Service Provider that provides access to the Internet to several customers located at the bottom of the diagram.

- In this made-up network, AS 65144 is well connected to several of the largest Internet carriers. In theory, AS 65144 pays different rates to send traffic via these carriers.
- AS 65144 has a bulk traffic link via AS 701. This path is of lower cost.
- The paths to make transit via AS 6939 (Hurricane Electric) and AS 3356 (LEVEL3) are more expensive.
- AS 65144 has a pricing system with regard its customers: premium and bulk.
- The customer AS 65002 pays premium rate and it runs BGP.
- The customer AS 65001 pays premium rate and it runs BGP although it only gets a default route from AS 65144.
- There is a number of customers that only have CPE devices (Customer Premises Equipment) such as modems, for example. These customers are simulated with VLANs.
- Some of the VLANs pay bulk prices and others (shown in the diagram) pay premium.
- Provision a solution to this problem by using a combination of LOCAL_PREF, MED, AS_PATH and COMMUNITY attributes.
- Test your solution by tracerouting from the customers to 12.0.0.0/8 in AS7018.

Chapter References

[1] R. Chandra, P. Traina, T. Li. Request for Comments RFC-1997. (1996.) BGP Communities Attribute.

[2] S Sangli, D. Tappan, Y. Rekhter. (2006). Request for Comments RFC-4390. BGP Extended Communities Attribute.

CHAPTER 8

Aggregation of Routes

The Internet IPv4 routing table had 467,037 entries at the moment that of the writing of this sentence (16-Nov-2013), but three days before, it had 466,466 entries, so 571 prefixes were added in such a short time. The table has not always grown, sometimes it has shrunk. For example, during the 1990s there was a great effort to reduce the size of the global Internet table by aggregating redundant prefixes. (on July 25, 2017 the table size is 645,628 IPv4 routes)

It is the best practice that only the necessary prefixes should be advertised from any autonomous system. However, not all domain administrators observe the best practices and then superfluous prefix details get propagated thorough the Internet. In this chapter, the principles of route aggregation or supernetting are explained. The principle being demonstrated is, that in certain topologies, having more details about the IP address allocation is not necessarily the best practice.

Aggregation of IP Route Addresses

Before getting into the configuration details of the routers, it is valuable to review the concept of aggregation of IP prefixes. The main idea of aggregation is to represent a group of common prefixes with a less specific prefix that covers only such prefixes.

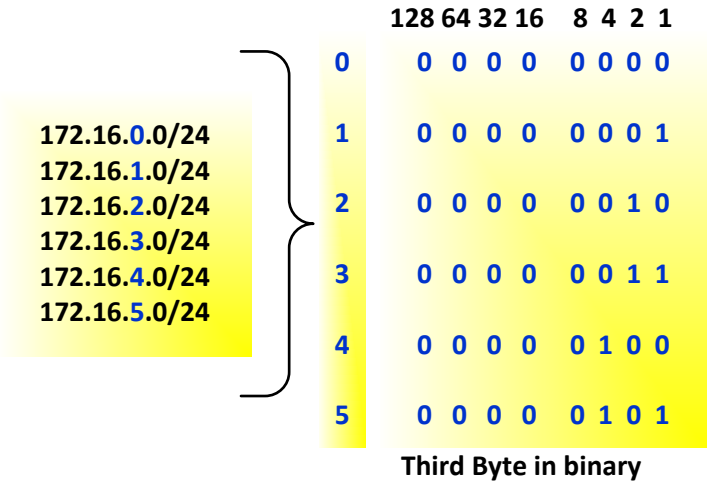


Figure 8.1

In the previous figure, there is a set of six IP addresses, all of them with the leading prefix 172.16. The difference appears in the third byte, so only that byte will be expanded into its binary bits for this explanation.

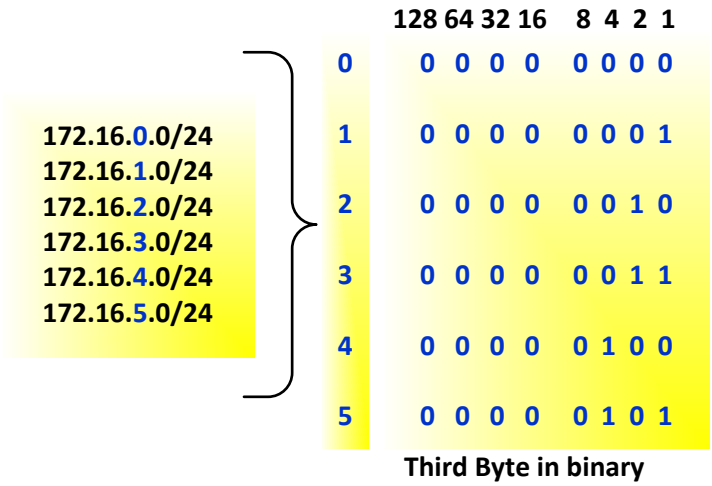


Figure 8.1

The examination of the binary combination reveals four possible subsets of binaries with common factors. Each one of the four subsets in the diagram can potentially be aggregated. The figure shows the positional value of each bit, but only the relevant bits are explained here.

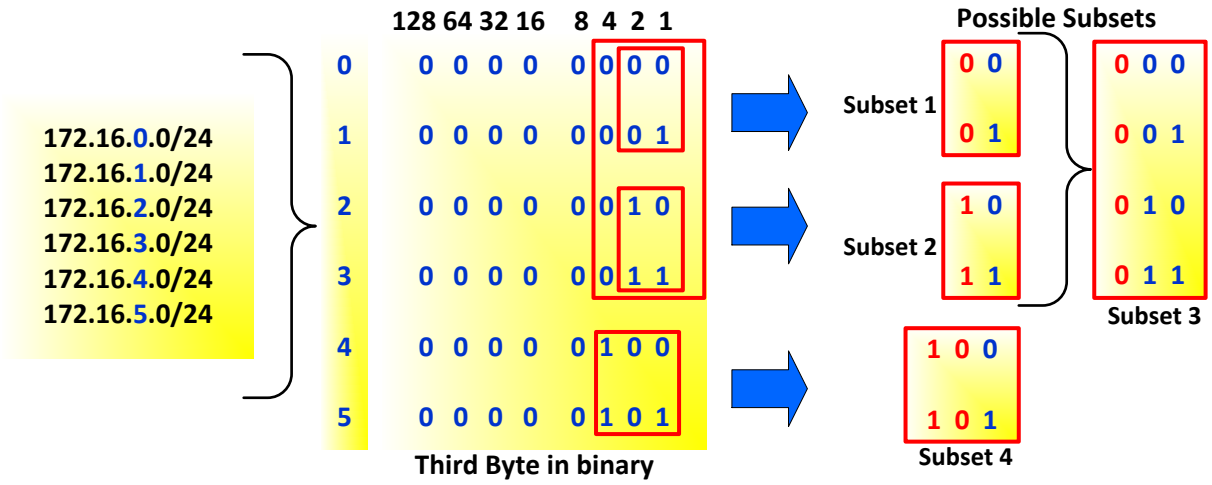


Figure 8.2

The key is to find common factors in an efficient way. For example, the common factor for subset 1 is a leading zero (actually 0000 000x for the whole byte.) A binary tree accompanied by the network mask is an efficient way to show this. The **x** in the nodes represents either value 0 or 1. Moving up one level of the tree makes the mask shorter by one, thus a supernet or aggregation is found.

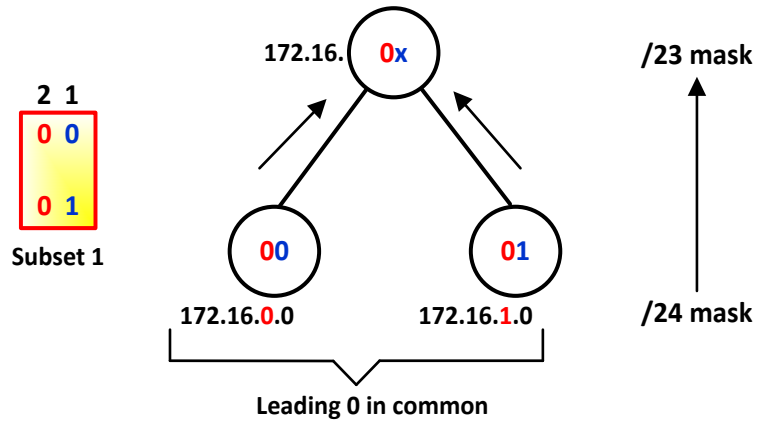


Figure 8.3

The first and second bytes are pre-set to 172.16 and the third byte is pre-set to zero and the matching mask is /23 according to the tree results. In consequence, the IP address 172.16.0.0/23 aggregates 172.16.0.0/24 and 172.16.1.0/24 and nothing else. That is the supernet or aggregation for subset 1.

Subset 2 is composed by 172.16.2.0/24 and 172.16.3.0/24 with a common leading factor of 1 (actually 0000 001x for the whole byte.) The position of such bit in the third byte is worth 2, so 172.16.2.0/23 is the aggregation of 172.16.2.0/24 and 172.16.3.0/24 and nothing else.

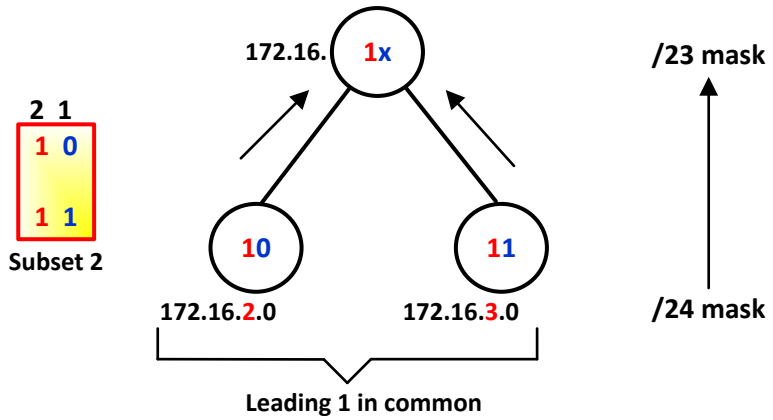


Figure 8.4

The two previous subsets can be aggregated by just one block (subset 3) by adding another top node to join the branches with common factors. The resulting mask is /22. The position of the leading bit is worth 4 but since the leading pattern is 0, then the third byte value is 0 also.

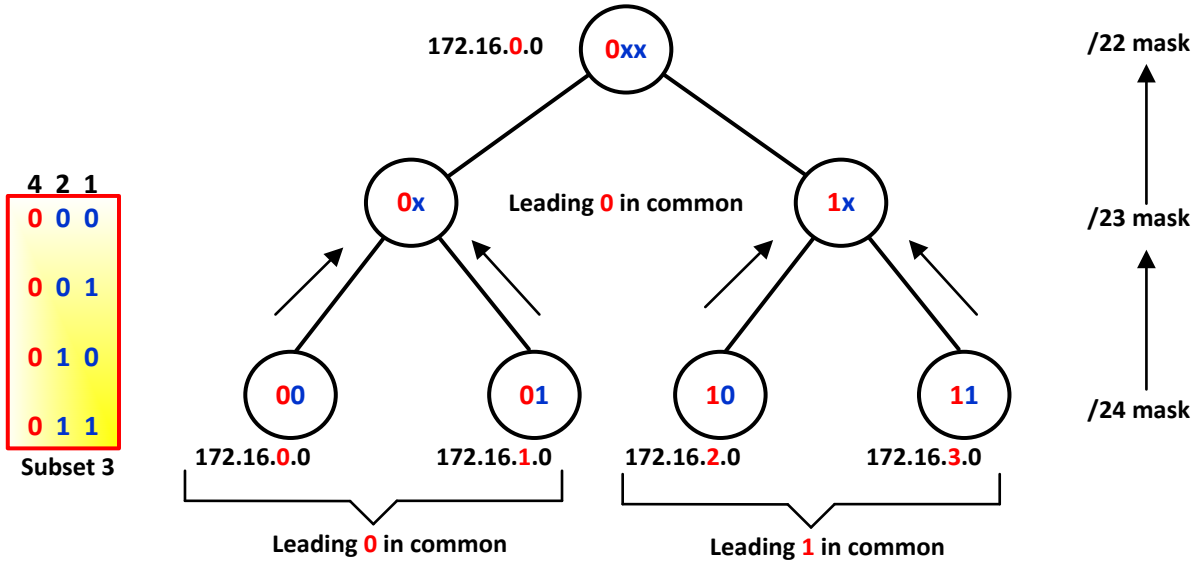


Figure 8.5

The aggregation 172.16.0.0/22 covers a range of four IP addresses from 172.16.0.0/24 to 172.16.3.0/24 and nothing else. So far, the aggregation choices are either two smaller aggregations of two addresses each or one larger aggregation that covers four addresses.

The last subset (4) contains two addresses with a leading pattern of 10 (actually 0000010x for the byte). The prefix 172.16.4.0/23 aggregates 172.16.4.0/24 and 172.16.5.0/24 and nothing more.

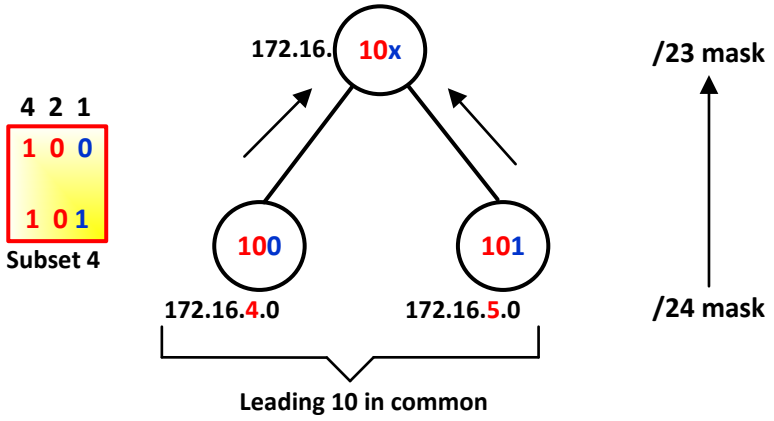


Figure 8.6

Notice that if the mask were made even shorter to /22 the following would happen:

	/22	/23	/24
172.16.4.0	1	0	0
172.16.5.0	1	0	1
172.16.6.0	1	1	0
172.16.7.0	1	1	1

Figure 8.7

A shorter mask, by only one bit, would cover two more addresses (172.16.6.0 and 172.16.7.0) which are not part of the assigned addresses, but they must belong to another assignment. It is necessary to stress this concept because in the Internet, missing the mask by just one bit could have serious consequences. For example, the following figure shows three autonomous systems with their respective address spaces.

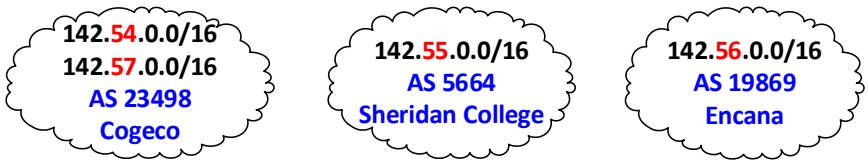


Figure 88

These organizations are completely unrelated to each other, so their address spaces must not be aggregated. In the case of Cogeco, the two address spaces cannot be aggregated either because they are not contiguous so there is no possible combination that will take only these two prefixes. Even more, let's imagine that the network administrator in Cogeco made a mistake by one bit in the mask and advertise 142.54.0.0/15 instead of /16 that will take the address space of Sheridan also and that might cause traffic disruptions. The lesson here is to really double check that the aggregation is only covering what is intended and nothing more.

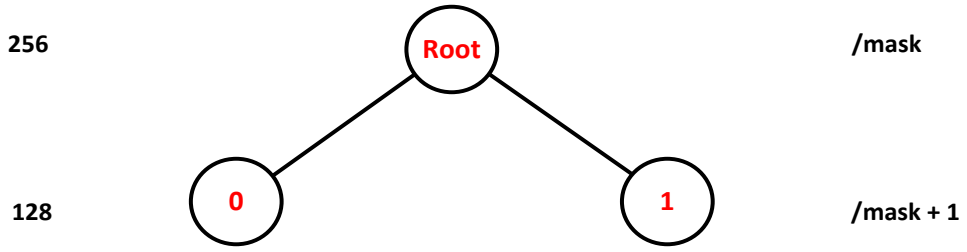
A Technique to Allocate Address Space

A common task in the operations of an Internet Service Provider is the provisioning of Internet access for a customer's routing domain which includes the physical connection and the allocation of IP address space. This address assignment must be done systematically and ISPs typically have an internal organization that administers their address space. In this section, a system is proposed to orderly issue IP address spaces to customers that can be aggregated also. This system is based on a tree that represents all the possible combinations that are available with the 8 bits of one byte. Next is the process, step by step, of how to build the tree and how to use it.

Step 1) Place a root node with the number 256 beside and a mask in slash format. The meaning of this step is that one byte can yield 256 combinations, but not all combinations will be shown here because the tree would be too long. The 256 combinations imply that the mask is either /8, or /16, or /24.

Step 2) If the stating prefix has a mask of /8, or /16, or /24, then place it beside the root. If not, wait until you reach the adequate mask level.

Step 3) Branch out the tree adding only two nodes underneath it. The node to the left is always worth binary zero (0) and the node to the right is always worth binary one (1). Since the address space has been divided equally, then place the number 128 beside the tree level (two nodes of 128 each account for the 256 addresses). Also, increase the mask by one.



- As a general rule, for every level going down, divide (halve) the previous address space and increase the mask plus one.

Step 4) Keep branching out following the previous rules, (two branches only per node, halve the previous space and increase the mask by one) like this:

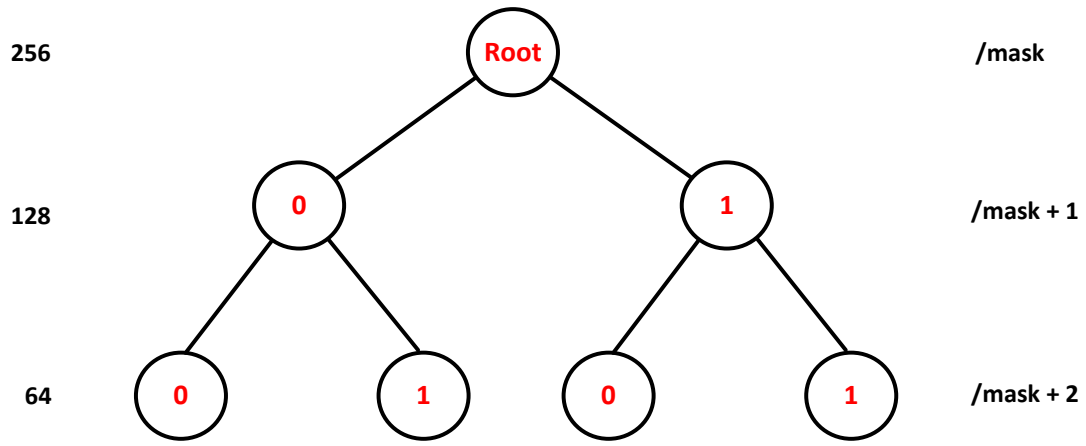


Figure 8.11

Step 5) Choose the address spaces as needed. For example, if you need two spaces of 64 and one of 128, just go to those levels and choose them. This case is represented in the next figure with the rectangles.

- As a general rule, begin from the left of the tree with the smallest spaces.

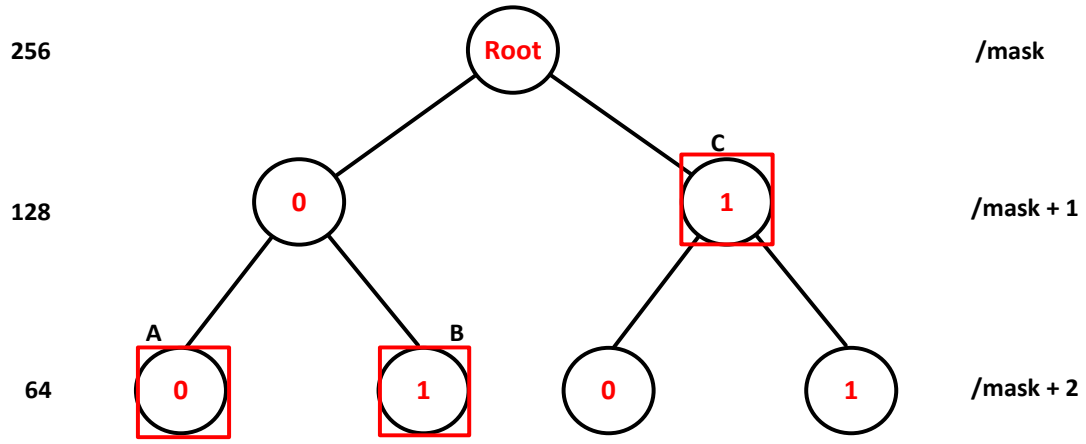


Figure 8.12

Step 6) Find the prefixes by adding to the root (an IP address) the values of the combinations of the nodes underneath. Basically, multiply the value inside the node by the number to the

right, then carry that value going down the tree branch. This is better explained with some examples.

Example 8.1

An address space 192.168.0.0/24 is available for allocation, take 64 addresses out of such space to be assigned to a network using the binary tree system.

The previous tree already has the setting needed to solve this problem. The squares, labeled from A to D, represent four possible solutions.

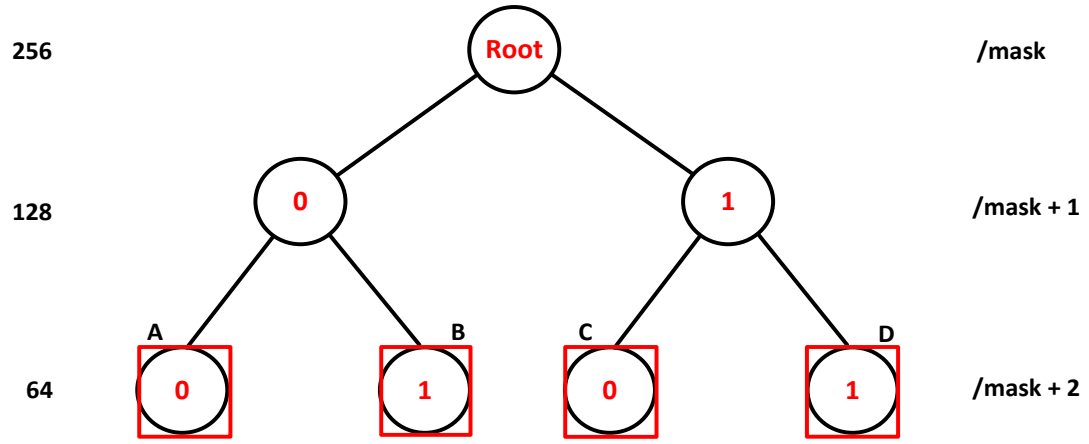


Figure 8.12

Let's examine the first option A. Starting from the root prefix 192.168.0, then going down the nodes, two binary zeroes are collected. The value of the prefix is 192.168.0.0 and the mask is /26.

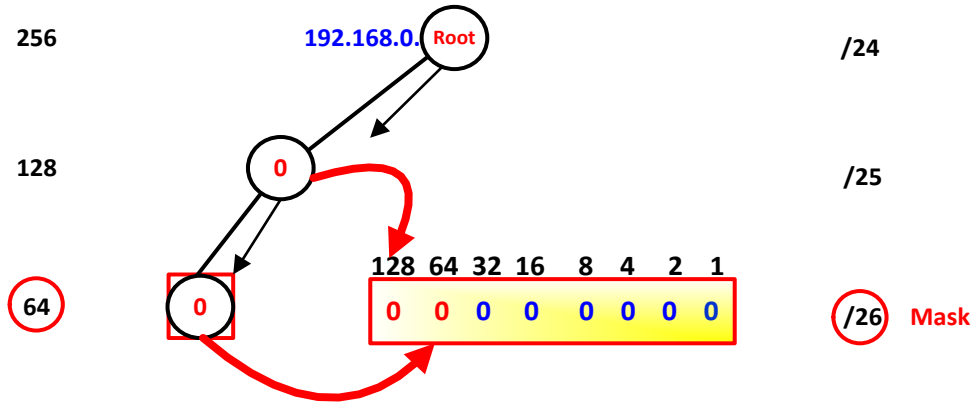


Figure 8.13

The second option B is represented by the following branch. The value 64 is attached to the prefix 192.168.0 creating 192.168.0.64 with a mask /26.

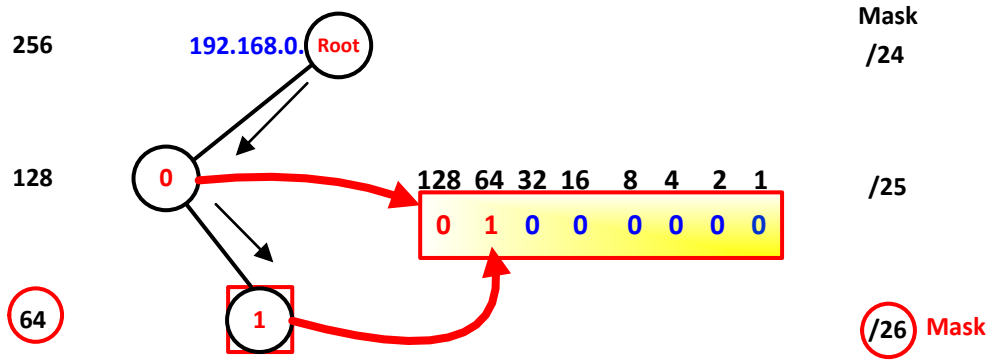


Figure 8.14

The third option would be 192.168.0.128/26.

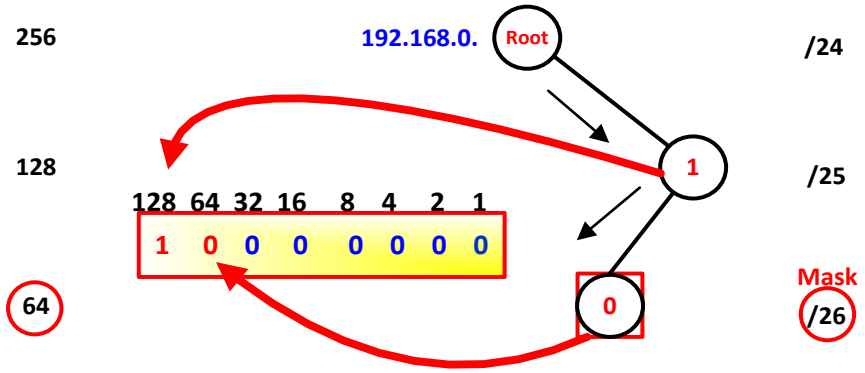


Figure 8.15

Finally, the last option would be 192.168.0.192/26.

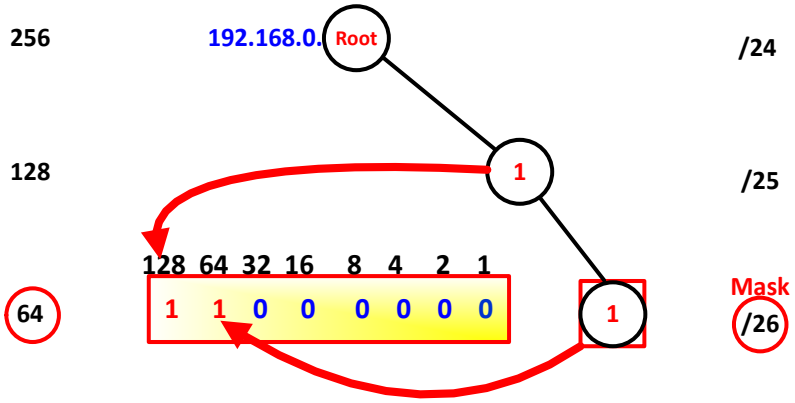


Figure 8.16

Any of the four previous options could be a valid answer to the initial problem. It is a matter of choosing one answer and the others will be left as reserved space for future allocations.

Although it is recommended that the allocation start from the left side since the prefixes are easier to visualize and to keep track of the space being allocated. The solution just explained is a case of IP address allocation of even space which tends to be easy to do for most network administrators. However, many networkers stumble into difficulty when trying to allocate space unevenly. There is even a jargon name for that, "variable length mask". The following examples concentrate in allocation of address space unevenly. This is where the binary tree system shows its advantages.

Example 8.2

The first example deals with a small address space so it is easier to visualize. The task is to allocate the address space 192.168.1.0/24 to the networks, represented by clouds in the diagram, according to the requirement of addresses.

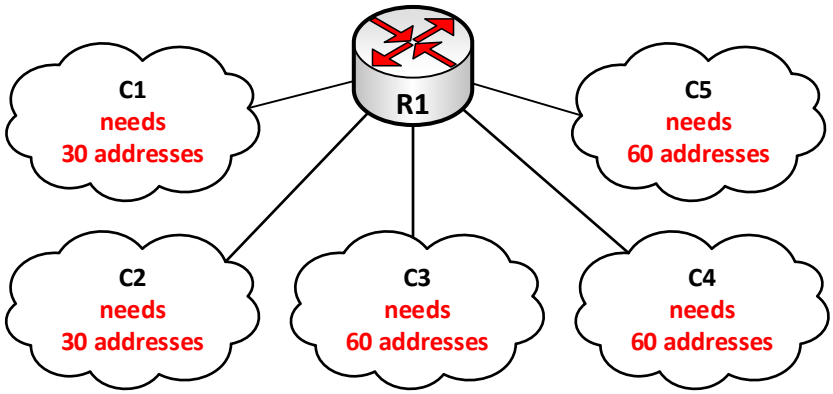
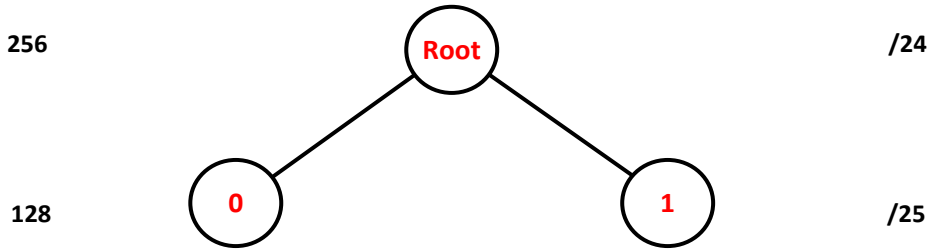


Figure 8.11

Let's set up the binary tree, first step is to set up the root:

The meaning of the root is that all the network prefixes will begin with 192.168.1. What is left to be determined are the values of the final byte and the corresponding mask. The mask /24 provides 254 IP addresses in the host space.

The second step is to branch out halving the previous address space.



The left node is value zero and the right node is activated with a one. Each of the nodes contains 128 IPv4 addresses with a mask /25. Let's continue expanding the tree until we reach the lowest level needed to solve the exercise.

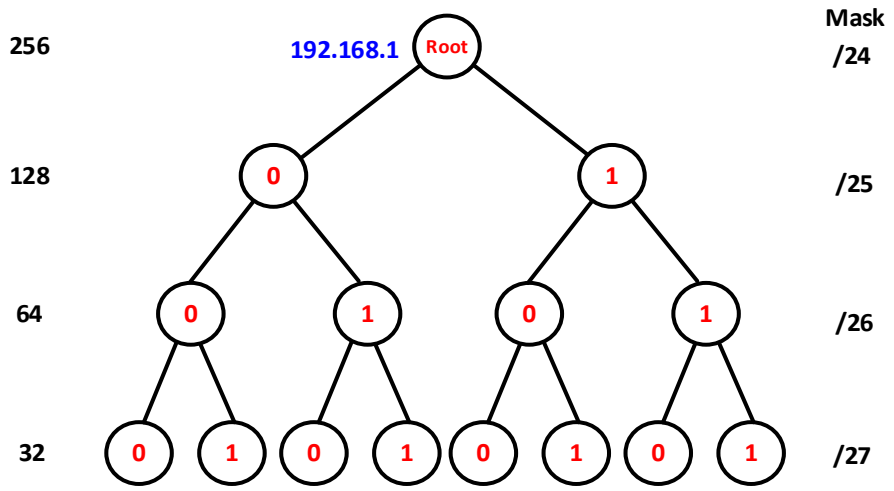


Figure 8.14

The smallest networks-clouds C1 and C2 need 30 addresses and the tree has already reached that address space size, so the tree can stop here and the excess of information can be trimmed. A possible solution to this problem is presented in the figure below. There are other possible valid combinations, but I recommend starting to allocate the smallest size networks from the left because the prefixes are easier to compute.

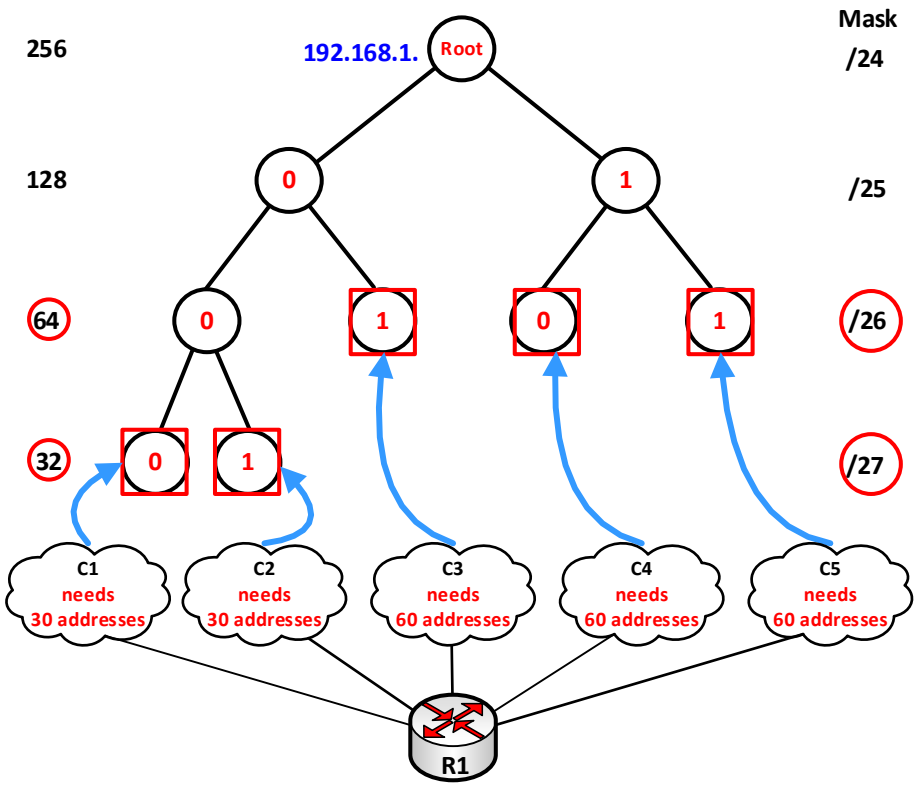


Figure 8.15

The nodes have been chosen according to the number of addresses required so now the step is to find the five network prefixes with their masks. This will be done in detail for a better explanation.

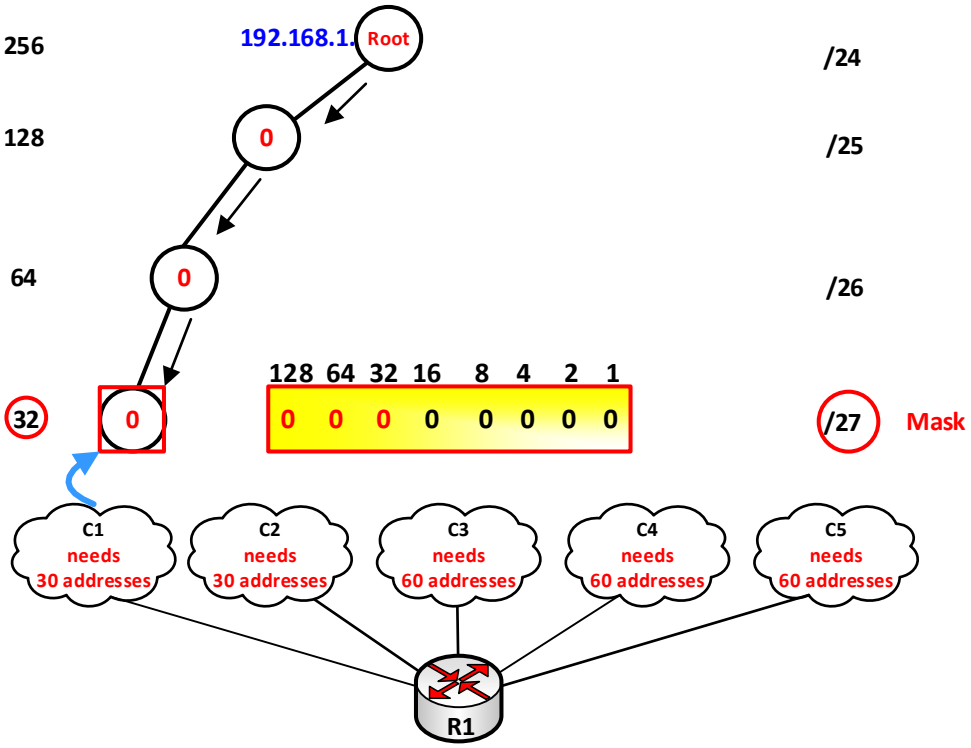


Figure 8.16

All the nodes from the root to the lowest level are zeroes so they do not add anything. That means, for the reader doing the binary counting, that the first three bits of the fourth byte are zeroes. The mask is shown in the circle to the right and it is /27. In consequence, the first prefix is 192.168.1.0/27 for a cloud of 32 addresses.

The second prefix is found next. The binary 1 value of the node has a significance of 32 as seen in both the binary tree and the binary counting of the byte. The preceding level bits are all zeroes. Then this value 32 is attached to the root prefix 192.168.1 to make the network 192.168.32.0/27.

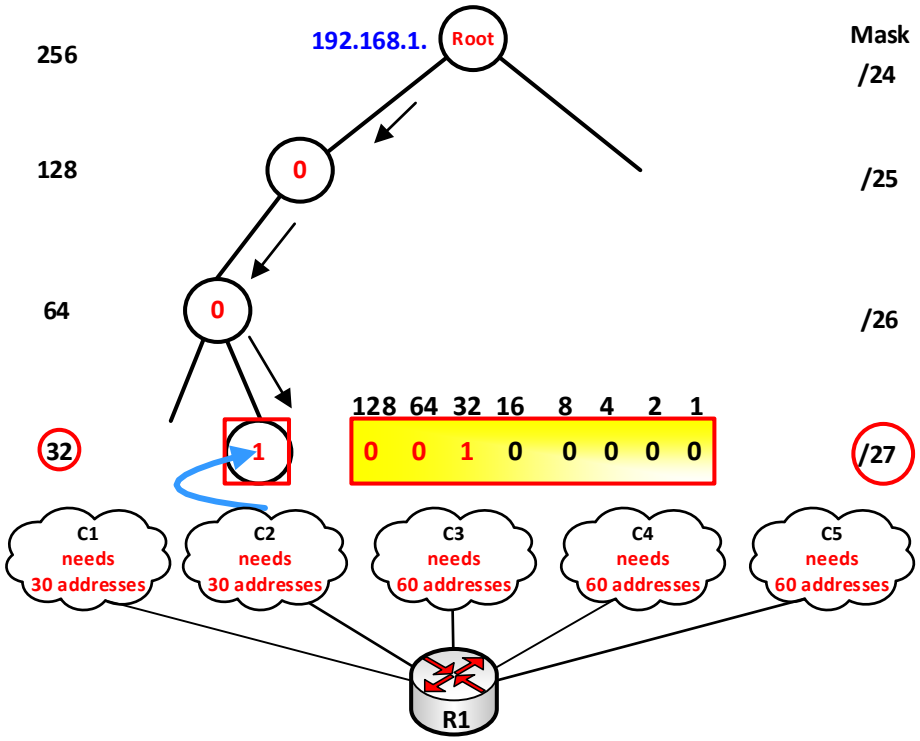


Figure 8.17

Cloud network C3 gets allocated the address space 192.168.1.64/26.

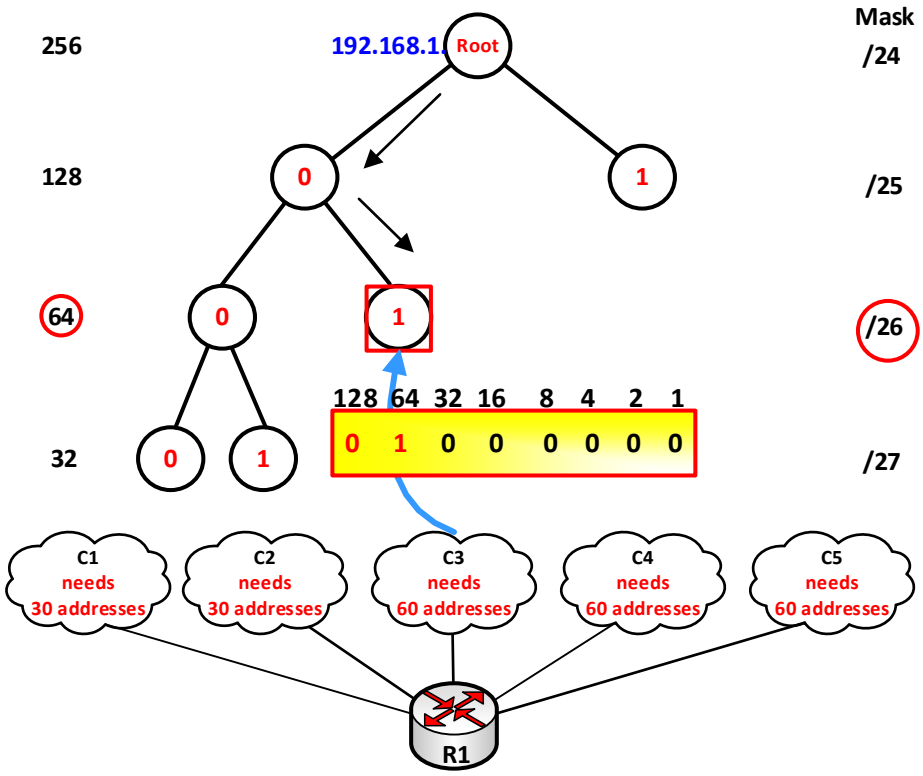


Figure 8.18

Cloud network C4 gets allocated the address space 192.168.1.128/26.

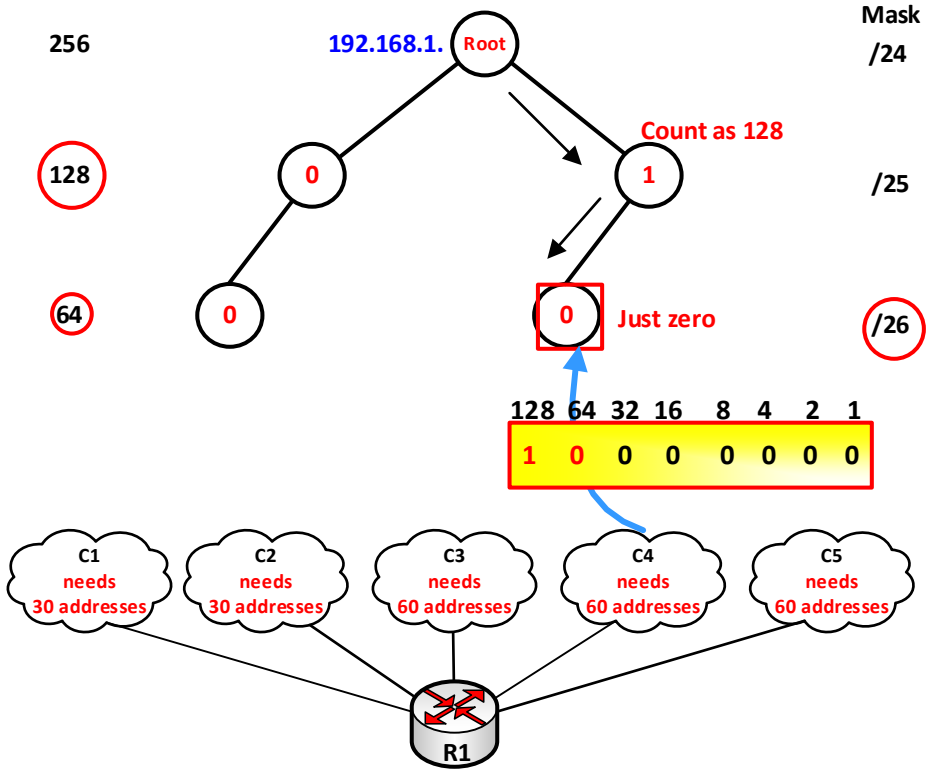


Figure 8.19

Finally cloud C5 network prefix is found to be 192.168.1.192/26.

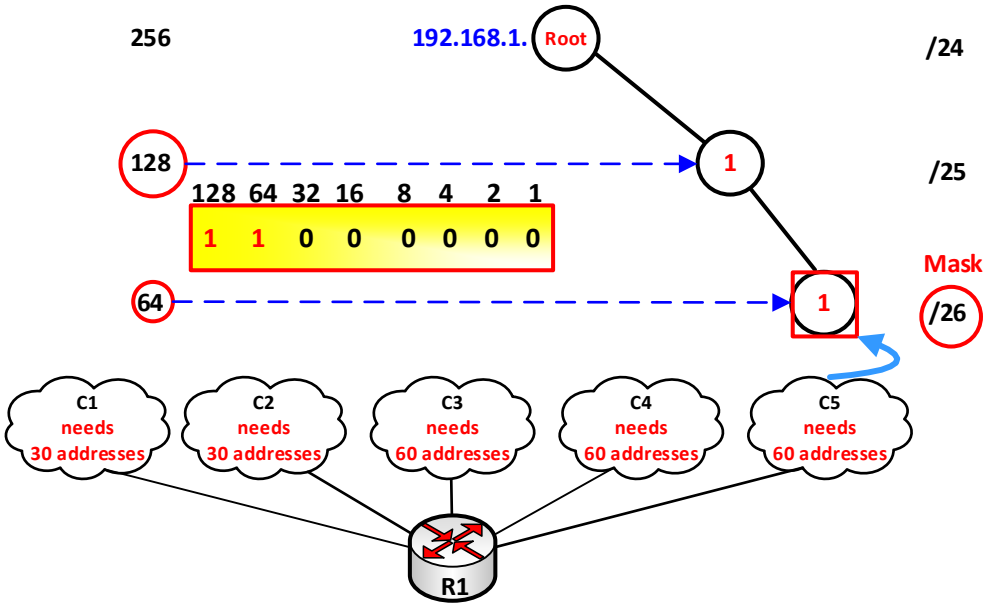


Figure 8.20

The complete solution is shown here.

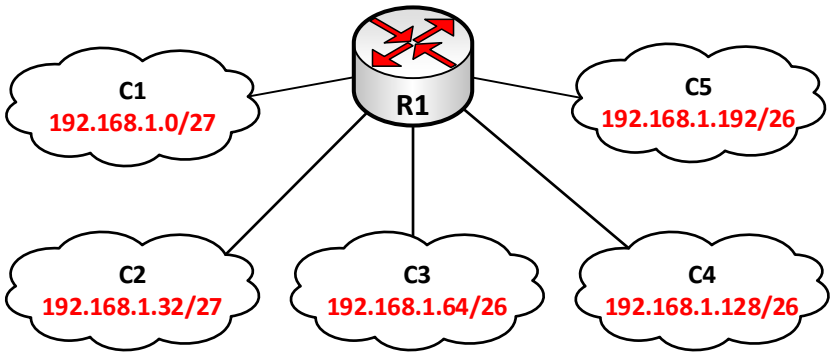


Figure 8.21

This example, so far, is a case of so called "sub-netting" because it is the easiest way to explain the functioning of the binary tree. Once that the functioning of the tree is understood, other examples will concentrate in just the aggregation of addresses. In any case, both of the so called "sub-netting" and "super-netting" are actually just address space allocation and the use of the binary tree is appropriated either way.

For example, let's aggregate 192.168.0.0/27 and 192.168.32.0/27 using the binary tree. The octagon represents the aggregation 192.168.1.0/26 of the two lower nodes.

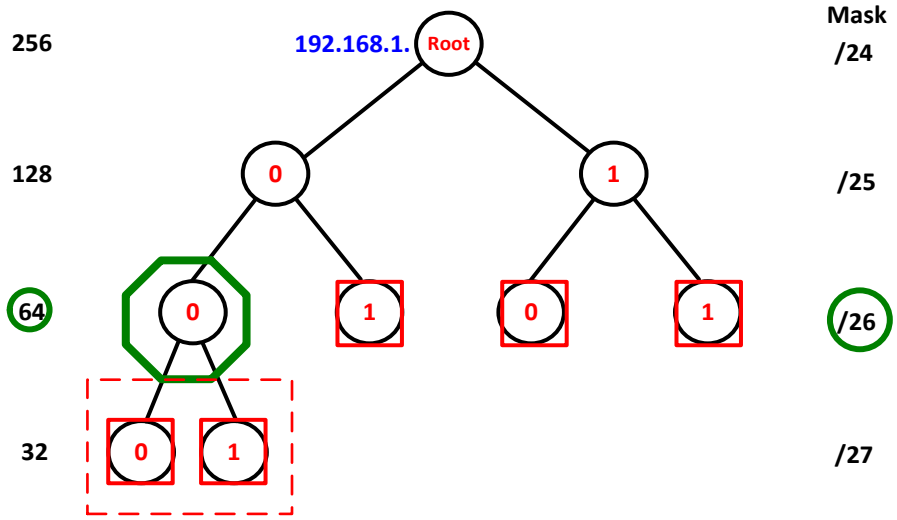


Figure 8.20

This octagon represents the aggregation 192.168.0.0/25 that covers all the address space underneath it.

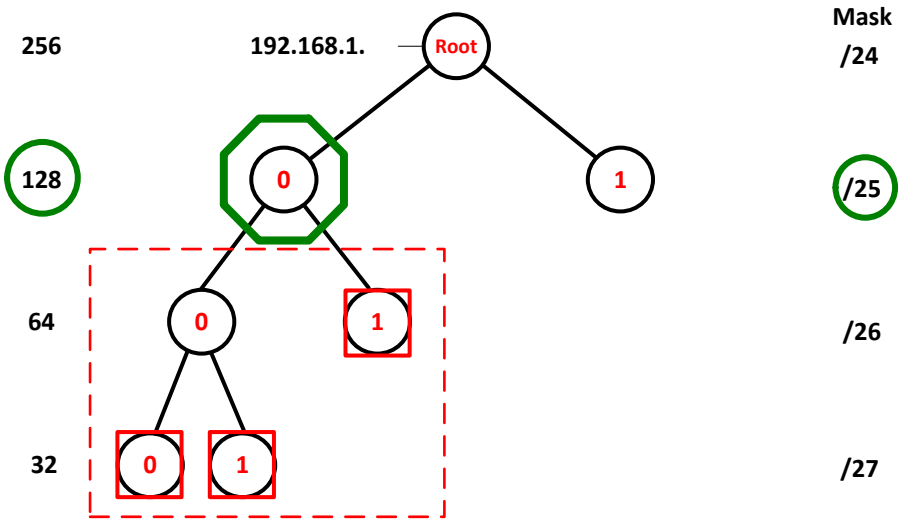


Figure 8.21

Finally, this octagon marks the position of the aggregation 192.168.1.128/25 which covers the address space of 192.168.1.128/26 and 192.168.1.192/26.

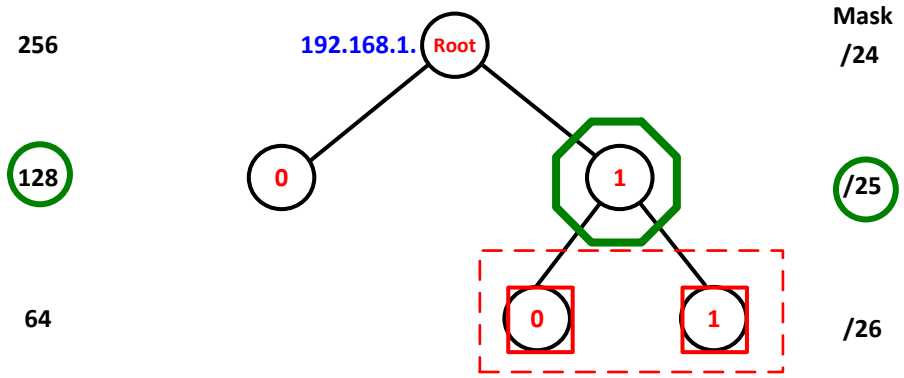


Figure 8.22

In conclusion, this example has shown that the binary tree can be used to find out the so called "sub-nettings" and also to find out the aggregations. The binary tree technique is a simple, clean, and systemic way to allocate IP address space.

Example 8.3

An Autonomous System 65001 connects to another AS. Internally, AS 65001 is a routing domain segmented into areas or levels. The IGP routing protocol is OSPF. However, other routing protocols such as ISIS or EIGRP would work fine as long as aggregations can be performed. Router R1 advertises in just one aggregation all the address space of AS65001 to the "other AS". Area border router R2 advertises into Area 0 the aggregation of Area 1 while ABR R3 advertises into Area 0 the aggregation of Area 2. Allocate 172.16.1.0/24 according to the address and aggregation requirements.

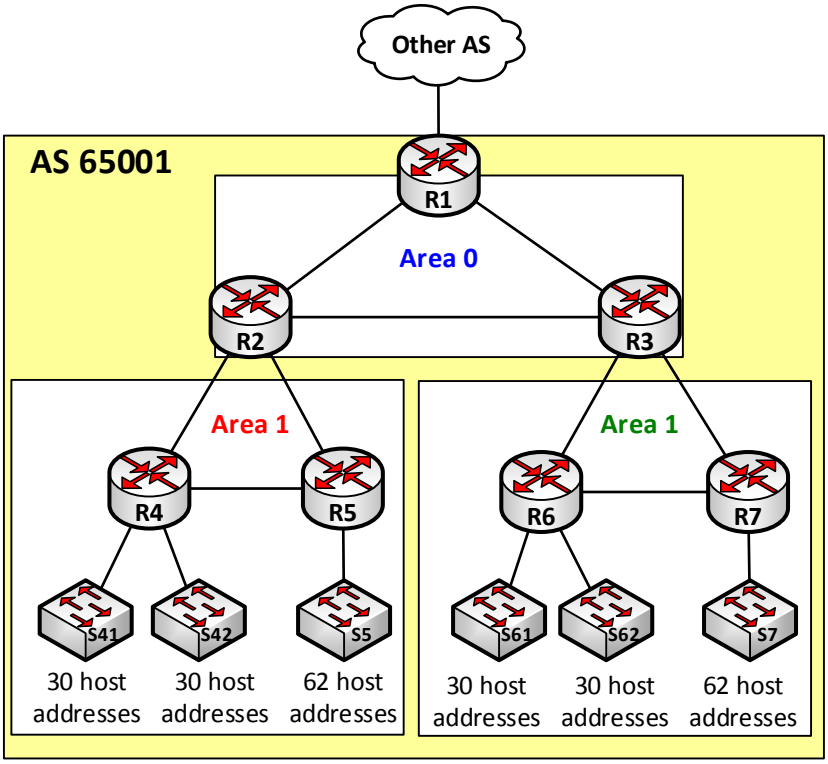


Figure 8.23

The solution for Area 1 is explained in the following diagram.

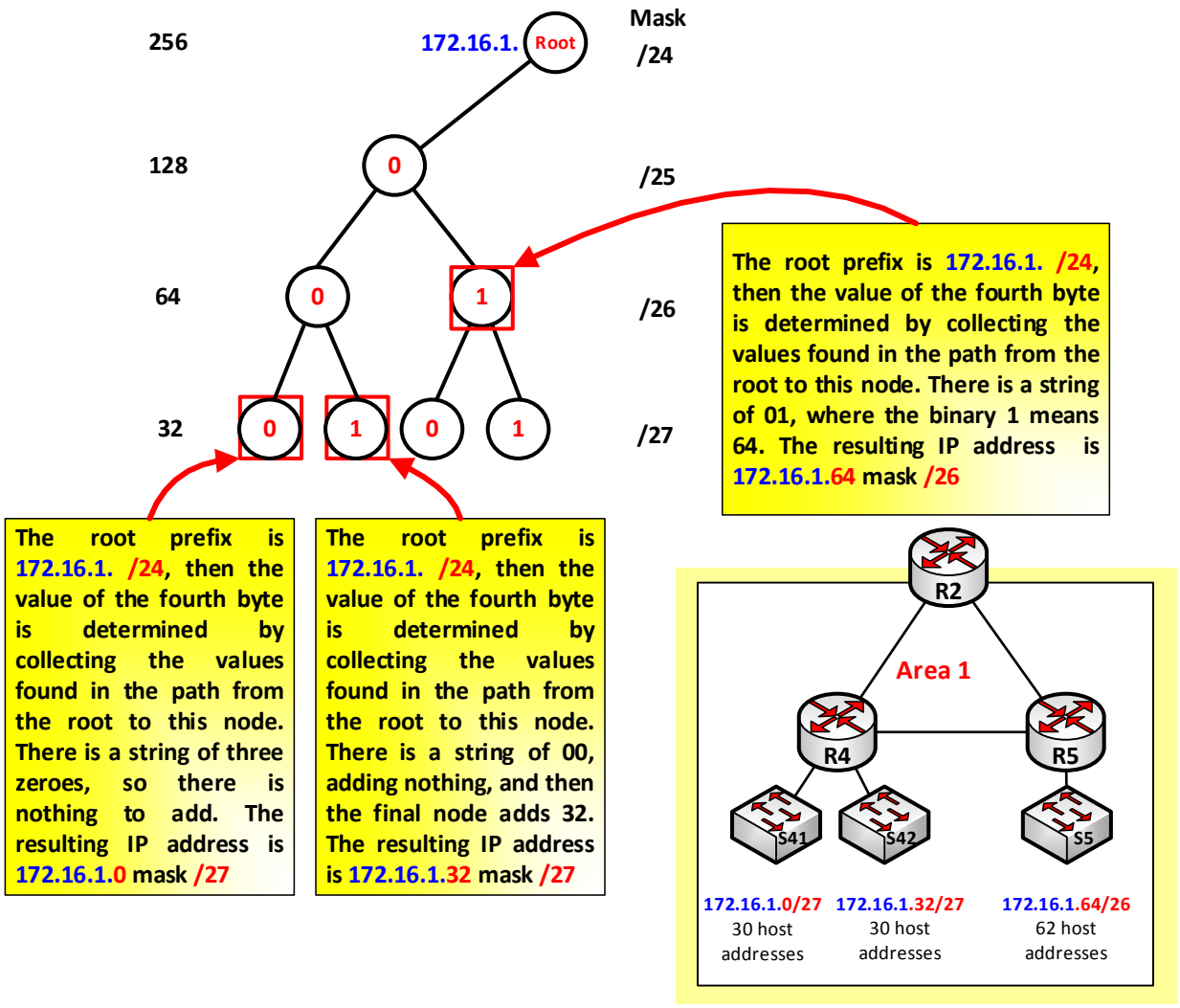


Figure 8.24

The solution for Area 2 is explained in the next figure.

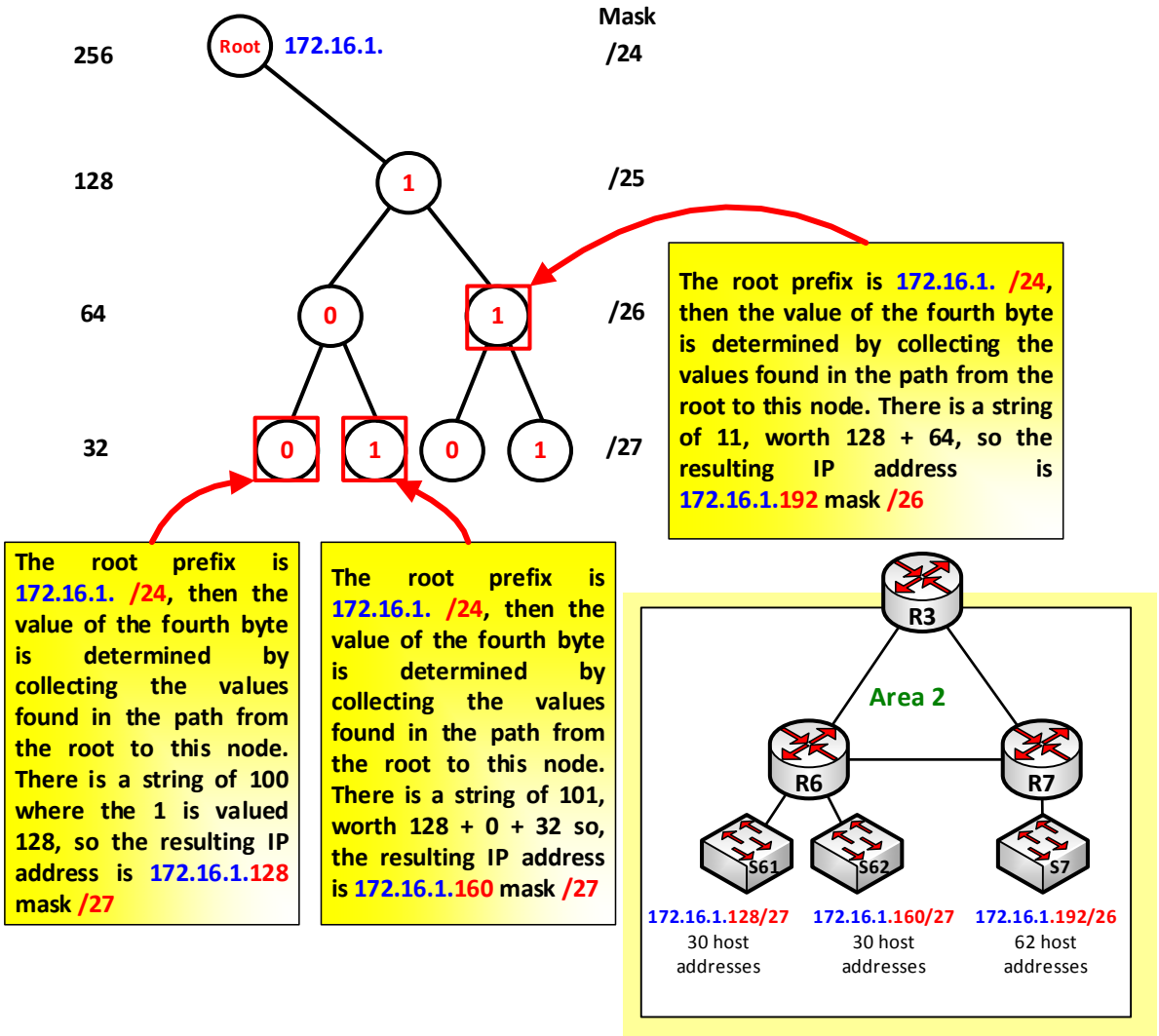


Figure 8.25

Finally, the binary tree helps to readily visualize the aggregations per area. This figure displays the complete solution. Area Border Router R2 advertises the aggregation 172.16.1.0/25, ABR R3 advertises 172.16.1.128/25 and ASBR R1 can advertise only 172.16.1.0/24 to other autonomous systems.

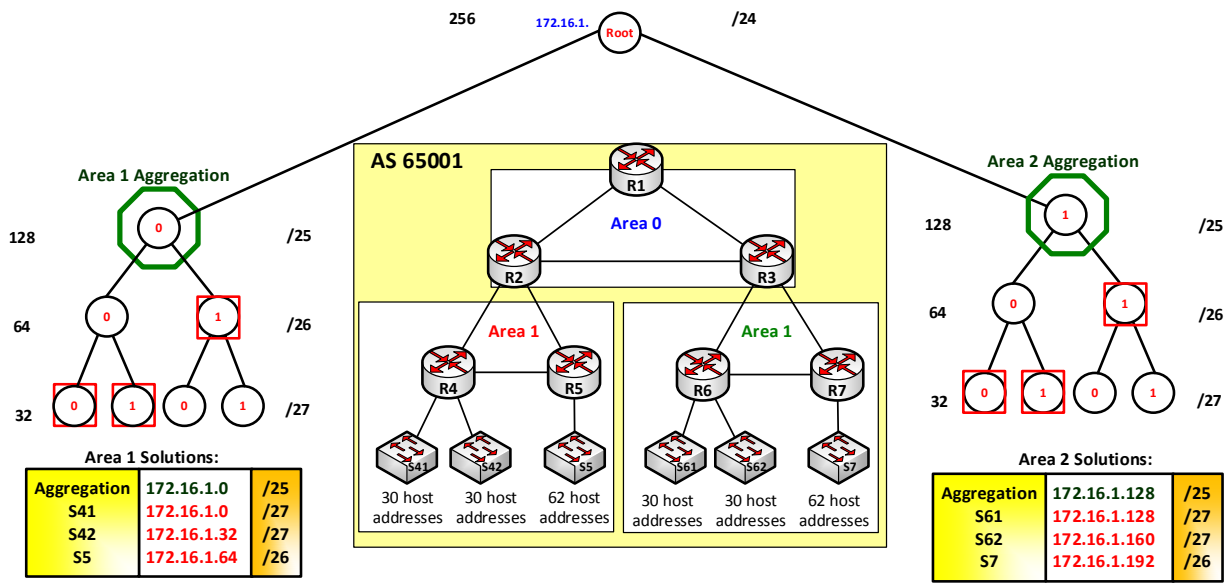


Figure 8.26

Example 8.4

Sheridan College owns 142.55.0.0/16 and it has three campuses, Oakville, Mississauga, and Brampton located in southern Ontario. A mask of /16 demarks a space of ($2^{(32-16)} = 2^{16}$) 65,536 IP addresses. Let's suppose that the network administrator estimates that the Trafalgar Campus needs about 30,000 addresses (a mask of /17), that the Brampton Campus needs about 16,000 addresses (a mask of /18), and that the Mississauga Campus needs about 8,000 addresses (a mask of /19).

Side Note: there are many ways to find the length of the mask, but the following formula can be a more rigorous approach. This equation is nothing more than 32 bits minus the host bit space, which yields the mask. (You do not need the 3.32 factor if your calculator has base-2 logarithm)

$$/mask = 32 - 3.32 \times \log_{10}(number\ of\ estimated\ IP\ addresses)$$

For example, for 8,000 addresses: $/\boldsymbol{mask} = 32 - 3.32 \times \log_{10}(8{,}000) = /\mathbf{19}$

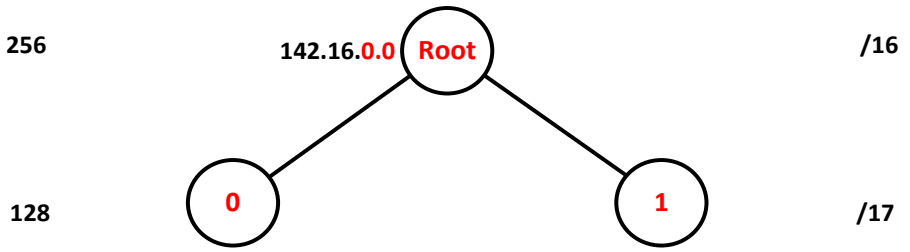
A mask of /19 means 32 − 19 = 13 bits of "host" space, which yields 2^{13} = 8192 prefixes. In summary:

Campus	Estimated Address Space	Mask	Possible Address Space
Mississauga	8,000	/19	$2^{(32-19)}$ = 8,192 addresses
Brampton	16,000	/18	$2^{(32-18)}$ = 16,384 addresses
Oakville	30,000	/17	$2^{(32-17)}$ = 32,768 addresses

Table 8.1

Now, once this preliminary work is done, let's proceed to find the specific address spaces for each site. First, let's set up the root for the binary tree:

Second, let' s branch out, halve the address space and increase the mask by one.



Keep doing the process until reaching the longest mask in the previous table (/19). There is no need to continue after that.

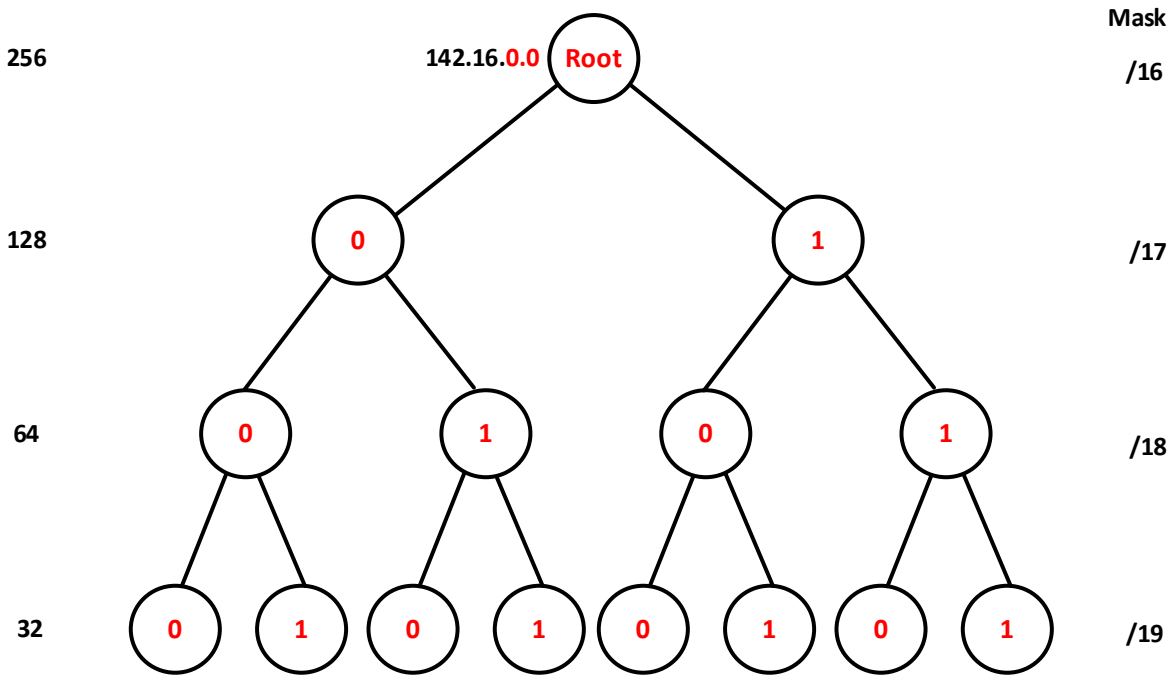


Figure 8.29

The tree is complete now so it can be simplified to display only the requirements:

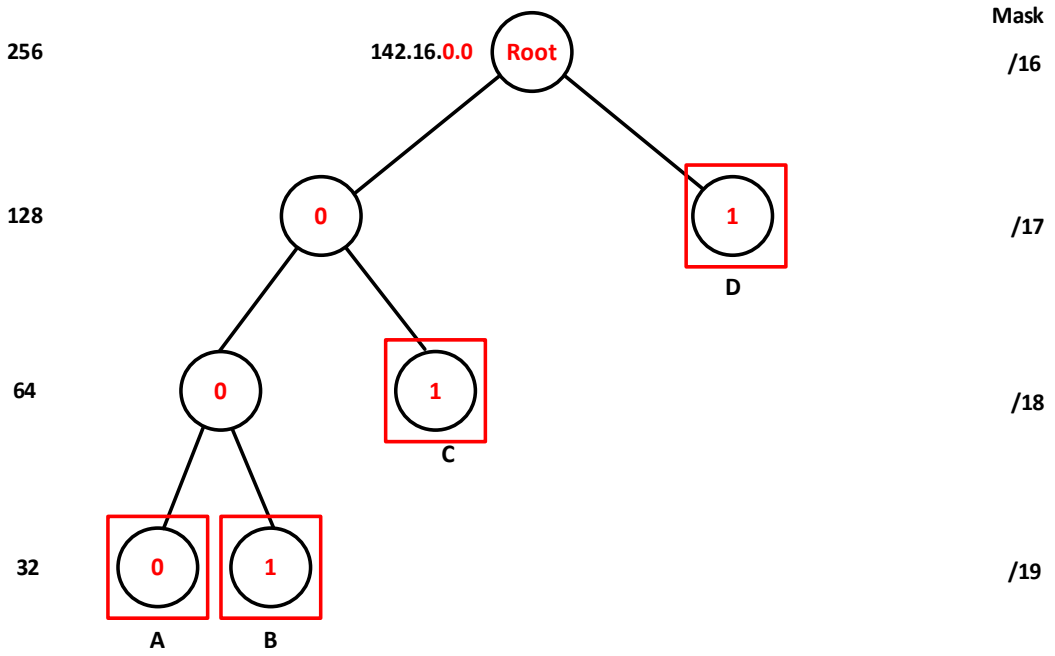


Figure 8.30

Letters have been added to the nodes for the purpose of the explanation. Either block A or B could be assigned to Mississauga Campus, let's take block B for example. The following diagram illustrates the principle behind the operation of the binary tree.

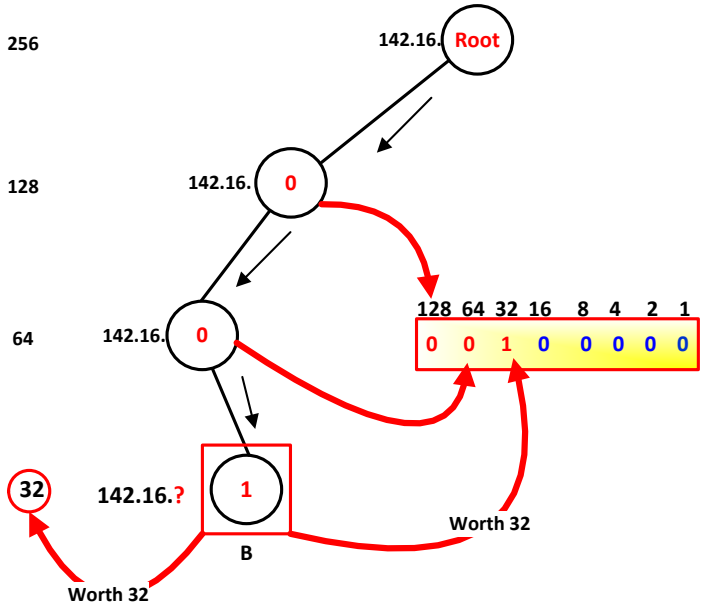


Figure 8.31

The binary tree is basically counting binary but graphically. The prefix root is 142.55., then by collecting the binary bits, they take on the positional value of the nodes. In the figure, the levels 128 and 64 add nothing because the nodes are set to zero, but the node B is worth 32 because there is a one inside the node. That results in the first prefix 142.55.32.0 with a mask /19 which is found by looking beside the level.

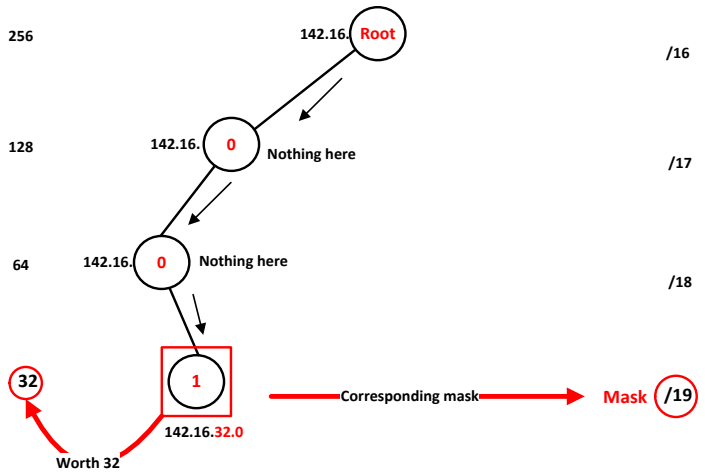


Figure 8.32

Similarly, node A could have chosen and would have been equally good to be allocated, for example:

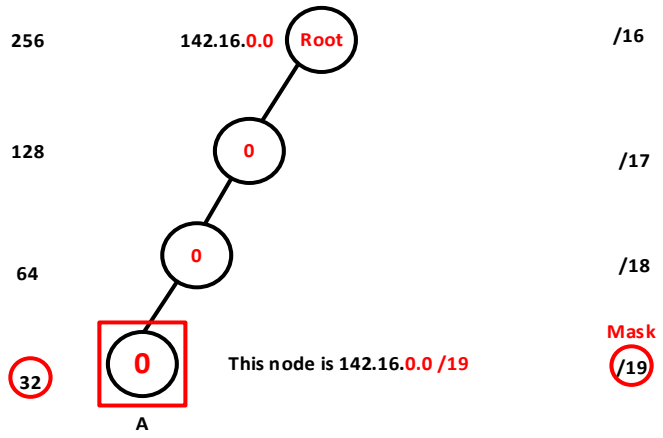


Figure 8.33

Either prefix 142.55.**0**.0/19 or 142.55.**32**.0/19 could be assigned to the Mississauga Campus. In any case, the one left out will be maintained as a reserve. Let's continue with the other Campuses Brampton and Oakville. For Brampton, the node labelled C has a value of binary 1

and it is at a level worth 64. The superior node contributes nothing to the prefix as the path with the arrows shows. The mask is /18, so the prefix for Brampton is 142.55.**64**.0/18.

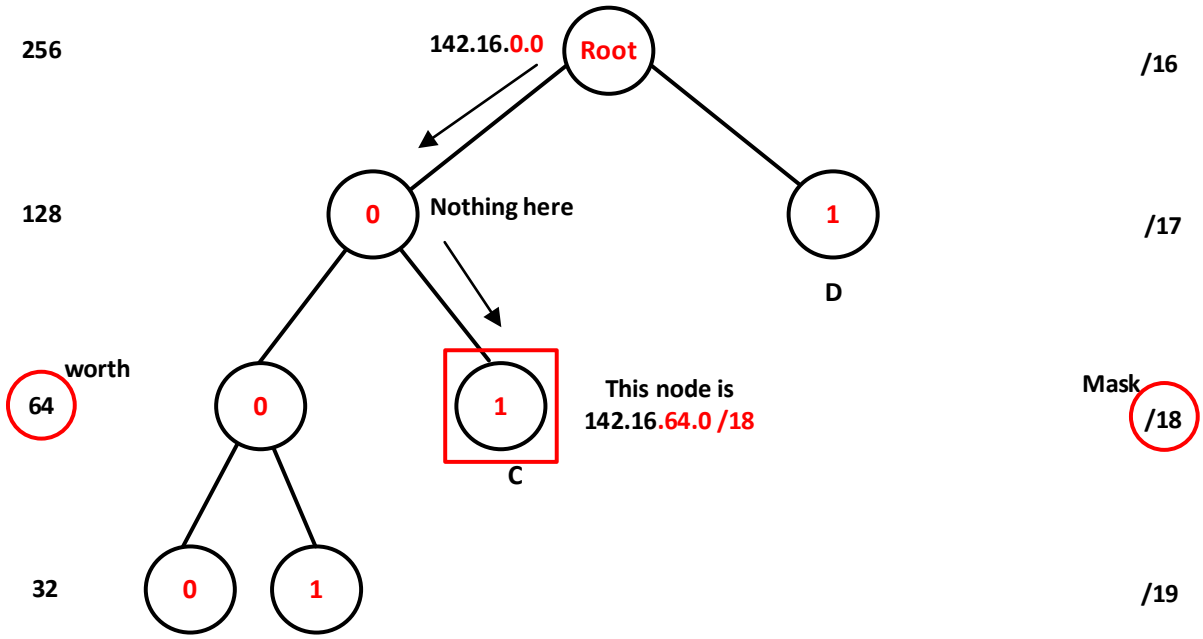


Figure 8.34

The last prefix is for the Oakville campus. The prefix 142.55.**128**.0/17 is found by adding the root prefix 142.55 to the value in the third byte which is 128 and the mask which is /17.

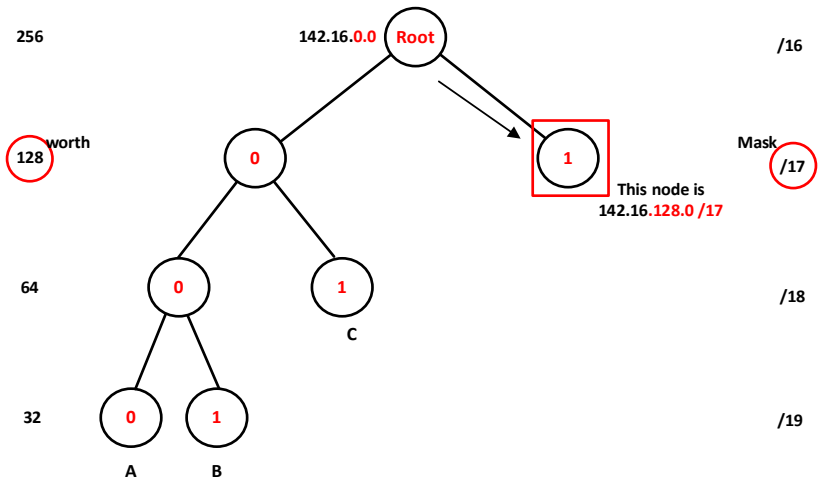


Figure 8.35

The following table and figure summarize the IP spaces for the three campuses. One /19 block is kept as a reserve.

Campus	Prefix	Mask	Address Space

Reserve	142.55.0.0	/19	$2^{(32-19)}$ = 8192 addresses
Mississauga	142.55.32.0	/19	$2^{(32-19)}$ = 8,192 addresses
Brampton	142.55.64.0	/18	$2^{(32-18)}$ = 16,384 addresses
Oakville	142.55.128.0	/17	$2^{(32-17)}$ = 32,768 addresses

Table 8.2

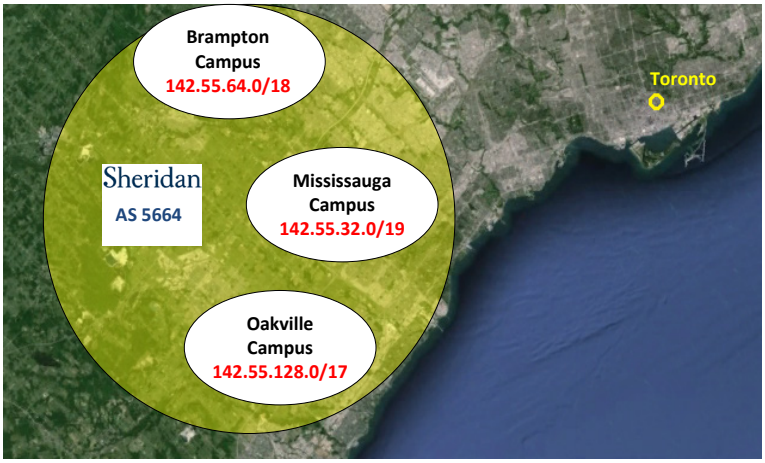


Figure 8.36

The College can advertise now the aggregations depending of the points of connection to other autonomous systems or it could advertise just one aggregation 152.55/16 that covers the whole space. That depends on the necessity of the administration of the routing domain. This type of decision will be covered in a section to come.

Example 8.5

Internet Service Providers tends to allocate address space using as a reference /24 mask networks. For instance, a customer that needs about 1000 IP addresses could be satisfy with the address space of four /24 networks (4 times 256 = 1024). How the customer assigns its address space internally, beyond /24, is not really relevant to the ISP.

Consequently, it is convenient to find a way to easily allocate /24 networks in blocks that can be aggregated. The binary tree can do exactly that. If the root mask is placed at /16, then the spaces that appear to the left of the tree can be interpreted as the number of /24 networks available. Let's explain that with an example. An ISP has the address block 24.128.0.0/16 available for its private customers as depicted in the next topology.

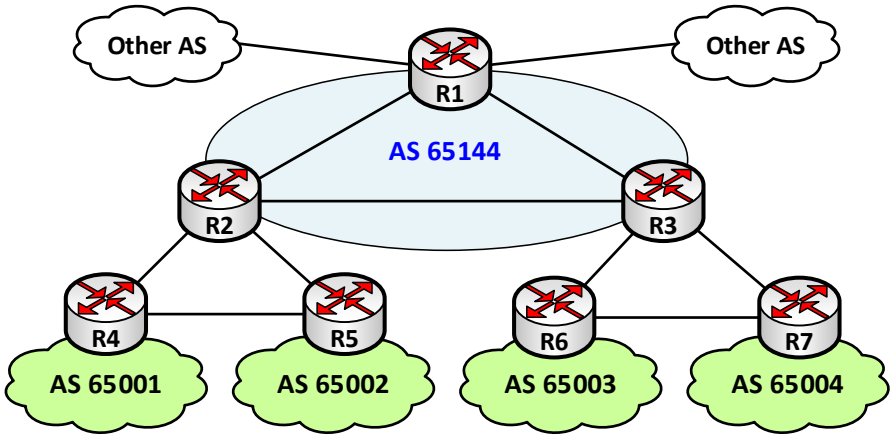


Figure 8.37

The customer requirements are displayed in this table.

Customer	Required Address Space	Equivalent /24 networks
AS65001	4,096 addresses	16 networks with mask /24
AS65002	4,096 addresses	16 networks with mask /24
AS65003	8,192 addresses	32 networks with mask /24
AS65004	16,384 addresses	64 networks with mask /24

Table 8.3

This binary tree shows a possible allocation. Half of the address space of 24.128.0.0/16 is allocated to the customers while the other half is kept as a reserve for future customers.

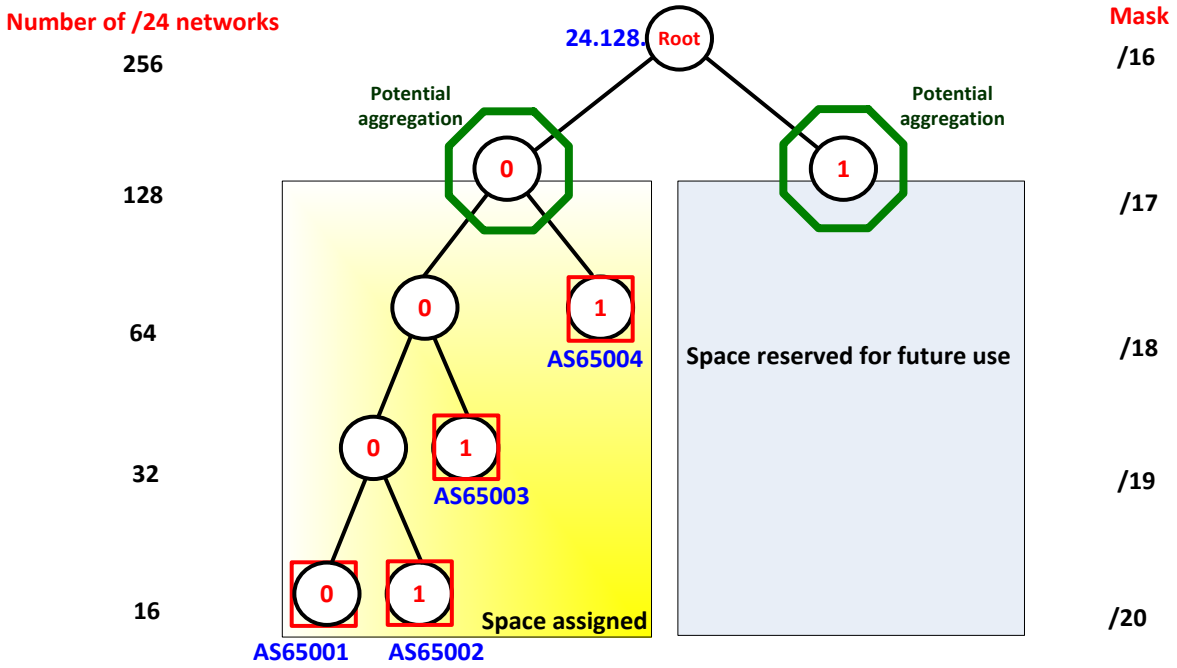


Figure 8.38

The solution is presented in this table.

Customer	Prefix	Equivalent /24 networks
Aggregation	24.128.0.0 /17	128 networks /24
AS65001	24.128.0.0 /20	16 networks /24
AS65002	24.128.16.0 /20	16 networks /24
AS65003	24.128.32.0 /19	32 networks /24
AS65004	24.128.64.0 /18	64 networks /24

Table 7.4

So far, a technique to administer IP address space has been explained. The next sections will deal with the implementation of aggregation in the routers. The binary tree is an organized way to allocate IP address space (for both aggregation and subnetting techniques). It removes guessing and if it is set up properly, it works all the time. There is an added benefit when doing aggregations, the upper node is always the aggregation address for the two nodes below so it makes visualization of address space easier.

Configuration of IPv4 Route Aggregation

The following topology will be used to demonstrate some route aggregation issues. The left side has Cisco IOS (R) routers and the right side Juniper (J) routers, so both ways of doing aggregation are explained. The left side has all the prefixes from the space 192.168.0.0/21 while the right side has the prefixes allocated from 172.16.0.0/16.

The network is designed in three tiers, starting from the top R1-AS1 and J1-AS10 act as a higher level of general connectivity. The second tier is made up by R2-AS2 and J2-AS20. These two AS would be Internet Service Providers ISPs to the customers AS3, AS4, AS30, and AS40.

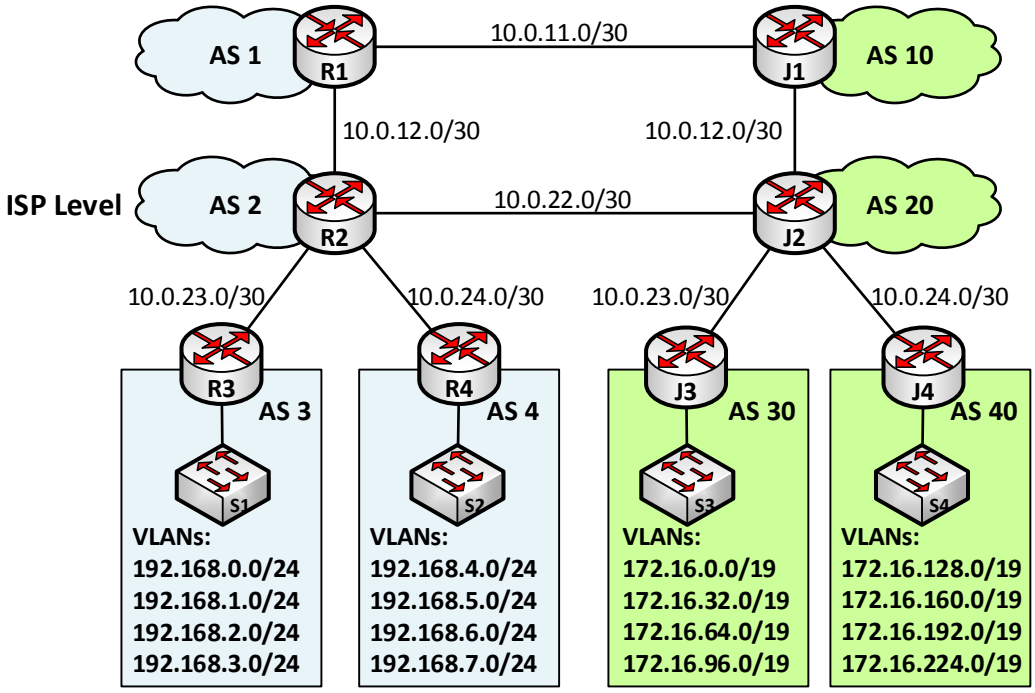
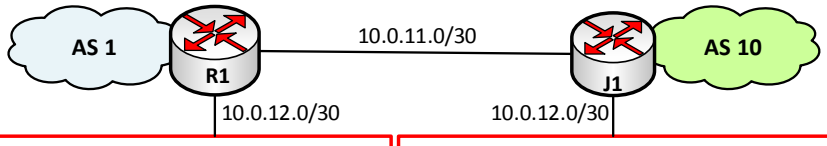


Figure 8.39

After the basic IP and BGP configuration is completed, the first step is to verify that the routing information is present in routers R1 and J1 at the top of the topology. In effect, the following figure shows that the BGP tables are consistent in both routers. There are 32 entries in the BGP database that will contribute with 16 prefixes to the IP routing table. The duplication of entries is caused by the redundant paths to the destinations.




```
R1#show ip bgp
  Network          Next Hop         Metric LocPrf  Weight  Path
* 172.16.0.0/19    10.0.12.2                               0  2 20 30 i
*>                 10.0.11.2                               0  10 20 30 i
* 172.16.32.0/19   10.0.12.2                               0  2 20 30 i
*>                 10.0.11.2                               0  10 20 30 i
* 172.16.64.0/19   10.0.12.2                               0  2 20 30 i
*>                 10.0.11.2                               0  10 20 30 i
* 172.16.96.0/19   10.0.12.2                               0  2 20 30 i
*>                 10.0.11.2                               0  10 20 30 i
* 172.16.128.0/19  10.0.12.2                               0  2 20 40 i
*>                 10.0.11.2                               0  10 20 40 i
* 172.16.160.0/19  10.0.12.2                               0  2 20 40 i
*>                 10.0.11.2                               0  10 20 40 i
* 172.16.192.0/19  10.0.12.2                               0  2 20 40 i
*>                 10.0.11.2                               0  10 20 40 i
* 172.16.224.0/19  10.0.12.2                               0  2 20 40 i
*>                 10.0.11.2                               0  10 20 40 i
* 192.168.0.0      10.0.11.2                               0  10 20 2 3 i
*>                 10.0.12.2                               0  2 3 i
* 192.168.1.0      10.0.11.2                               0  10 20 2 3 i
*>                 10.0.12.2                               0  2 3 i
* 192.168.2.0      10.0.11.2                               0  10 20 2 3 i
*>                 10.0.12.2                               0  2 3 i
* 192.168.3.0      10.0.11.2                               0  10 20 2 3 i
*>                 10.0.12.2                               0  2 3 i
* 192.168.4.0      10.0.11.2                               0  10 20 2 4 i
*>                 10.0.12.2                               0  2 4 i
* 192.168.5.0      10.0.11.2                               0  10 20 2 4 i
*>                 10.0.12.2                               0  2 4 i
* 192.168.6.0      10.0.11.2                               0  10 20 2 4 i
*>                 10.0.12.2                               0  2 4 i
* 192.168.7.0      10.0.11.2                               0  10 20 2 4 i
*>                 10.0.12.2                               0  2 4 i
```

```
felix@J1> show route protocol bgp terse
A Destination      P Prf   Metric 1   Metric 2  Next hop      AS path
* 172.16.0.0/19    B 170   100                  >10.0.12.2    20 30 i
                   B 170   100                   >10.0.11.1   1 2 20 30 i
* 172.16.32.0/19   B 170   100                  >10.0.12.2    20 30 i
                   B 170   100                   >10.0.11.1   1 2 20 30 i
* 172.16.64.0/19   B 170   100                  >10.0.12.2    20 30 i
                   B 170   100                   >10.0.11.1   1 2 20 30 i
* 172.16.96.0/19   B 170   100                  >10.0.12.2    20 30 i
                   B 170   100                   >10.0.11.1   1 2 20 30 i
* 172.16.128.0/19  B 170   100                  >10.0.12.2    20 40 i
                   B 170   100                   >10.0.11.1   1 2 20 40 i
* 172.16.160.0/19  B 170   100                  >10.0.12.2    20 40 i
                   B 170   100                   >10.0.11.1   1 2 20 40 i
* 172.16.192.0/19  B 170   100                  >10.0.12.2    20 40 i
                   B 170   100                   >10.0.11.1   1 2 20 40 i
* 172.16.224.0/19  B 170   100                  >10.0.12.2    20 40 i
                   B 170   100                   >10.0.11.1   1 2 20 40 i
* 192.168.0.0/24   B 170   100                  >10.0.12.2    20 2 3 i
                   B 170   100                   >10.0.11.1   1 2 3 i
* 192.168.1.0/24   B 170   100                  >10.0.12.2    20 2 3 i
                   B 170   100                   >10.0.11.1   1 2 3 i
* 192.168.2.0/24   B 170   100                  >10.0.12.2    20 2 3 i
                   B 170   100                   >10.0.11.1   1 2 3 i
* 192.168.3.0/24   B 170   100                  >10.0.12.2    20 2 3 i
                   B 170   100                   >10.0.11.1   1 2 3 i
* 192.168.4.0/24   B 170   100                  >10.0.12.2    20 2 4 i
                   B 170   100                   >10.0.11.1   1 2 4 i
* 192.168.5.0/24   B 170   100                  >10.0.12.2    20 2 4 i
                   B 170   100                   >10.0.11.1   1 2 4 i
* 192.168.6.0/24   B 170   100                  >10.0.12.2    20 2 4 i
                   B 170   100                   >10.0.11.1   1 2 4 i
* 192.168.7.0/24   B 170   100                  >10.0.12.2    20 2 4 i
                   B 170   100                   >10.0.11.1   1 2 4 i
```

Figure 8.40

The routing information in the BGP tables of R1 and J1 does not represent a particular benefit in this physical topology since the destination networks are all located in the customer ASes. On the contrary, the details of the IP address allocation from the customers AS3, AS4, AS30 and AS40 are just cluttering the routing tables of the top routers. Besides, the excessive entries demand more router resources such as memory, CPU cycles and processing time. This network topology would benefit if some aggregation were implemented in the right places. Let's do just that step by step.

Customer Advertises Aggregation plus Details

Ideally, autonomous systems would only advertise the necessary IP prefixes to accomplish connectivity. So in the example topology, AS3 will aggregate its prefixes, as much as possible, to simplify the work at the higher level networks. The following diagram shows an aggregation being done in the Cisco router R3. The Cisco's "**aggregate-address**" command is issued under the configuration of BGP and its effect is observed in router R1. (In the meantime, the details about the 172.16 prefixes have been removed to simplify the analysis of the data).

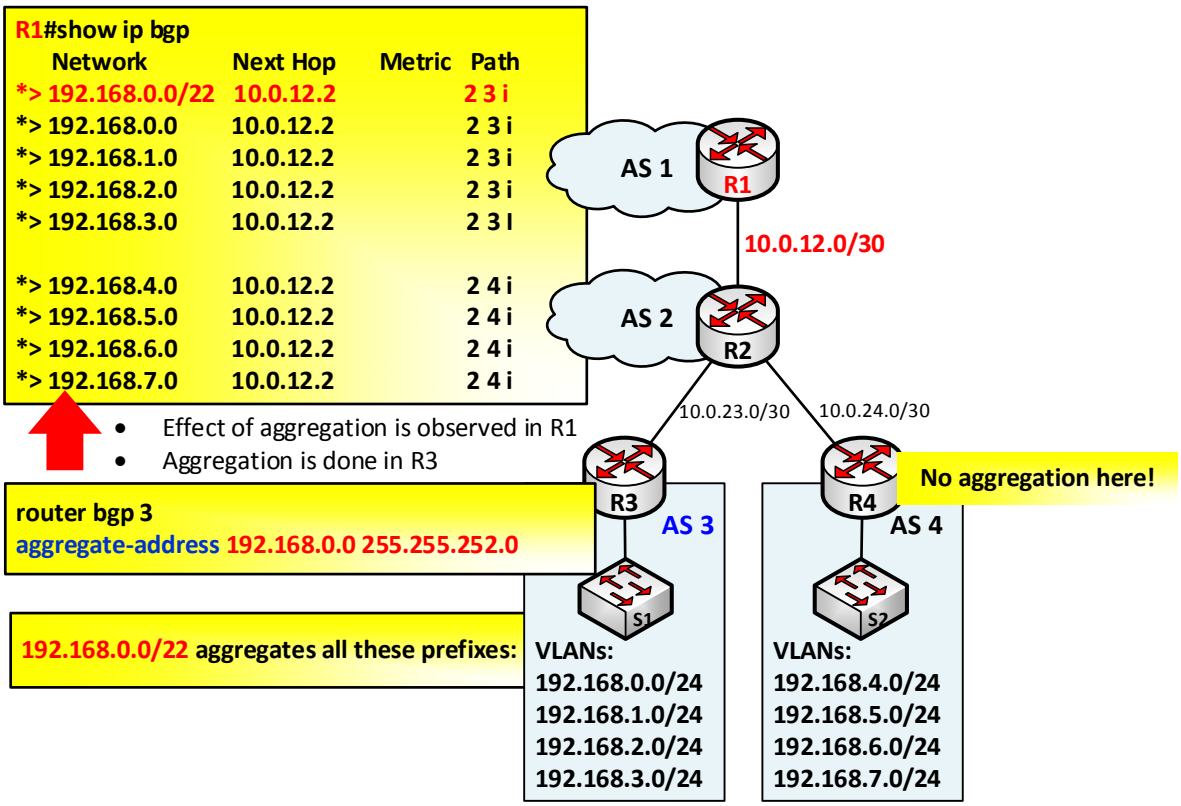


Figure 8.41

Router R1 is not gaining anything with the aggregation. In fact, its BGP table increases by one entry so what is the point of doing the aggregation? Sometimes, as per agreement, the lower levels send the **aggregation plus prefix details** to let the upper levels decide what is going to be either filtered out or advertised or match with a community value. In such model, R1 or R2 would have to implement a policy to do so.

The model of aggregation plus details would typically be implemented in a network that needs to advertise a supernet that covers all the generic traffic network routes plus the specific detail about an important network; for example, the subnet where the application servers are located in a data centre.

However, in the current physical topology, advertising details does not yield a benefit, quite the opposite. Later, a different network will be used to further explain this model. But the point has been made that the "aggregate-command" alone might not be enough. Let's move on to a simpler aggregation model.

Customer Advertises Aggregation Only

Cisco Implementation

In this case, the source of the routing information advertises an aggregation without any other details. This is typically the situation when the physical connectivity to the other peers is simple like in the example network.

Now, Cisco router R3 performs the aggregation with the "**summary-only**" keyword. This sub-command controls that only the aggregation is advertised without the specific prefixes covered by such supernet.

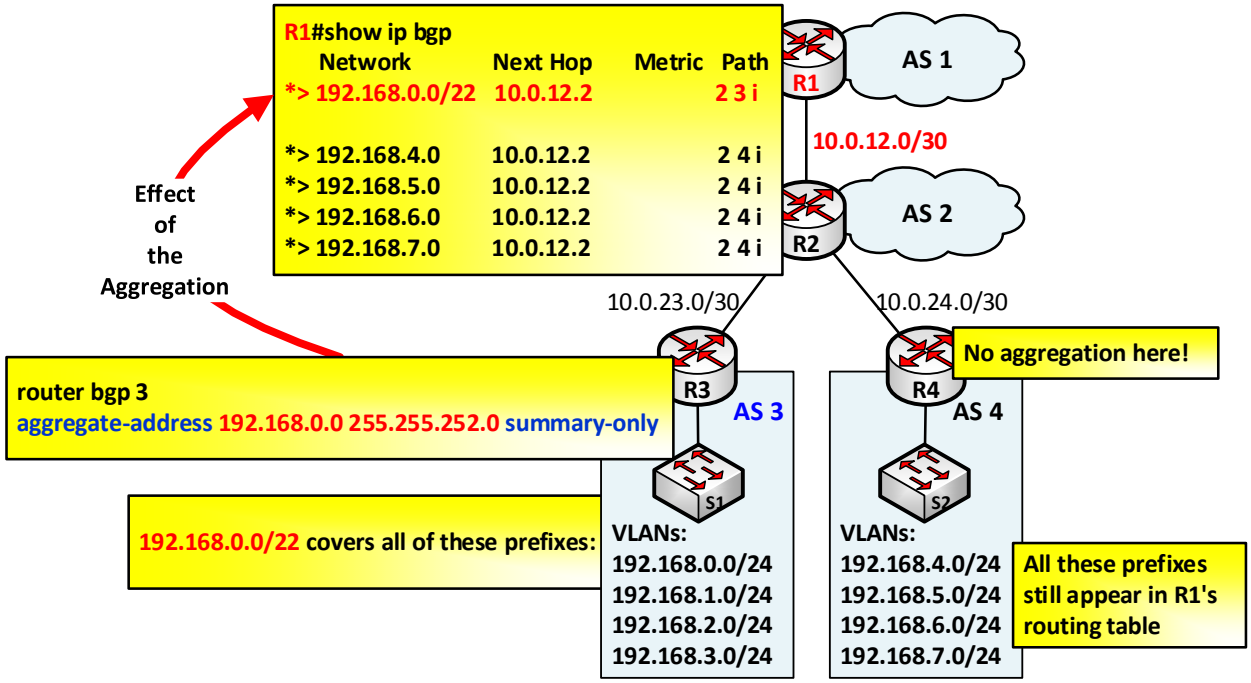


Figure 8.42

Router R3 sends the aggregation 192.168.0.0/22 to cover the block of prefixes from 192.168.0.0/24 to 192.168.3.0/24 as the binary tree shows. This aggregation-only suppresses the more specific prefixes.

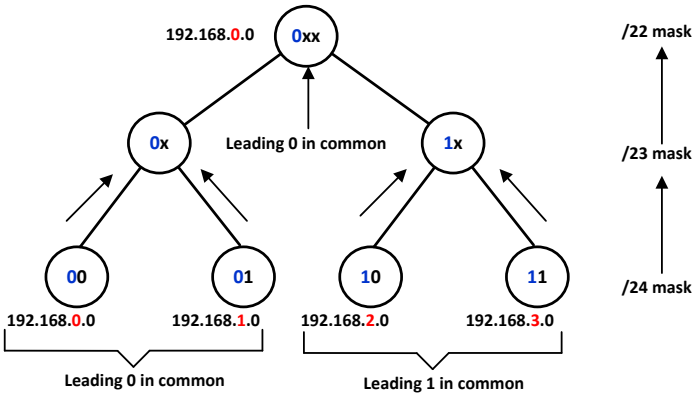


Figure 8.43

The routing table reduction is even more evident when aggregation is also implemented in R4.

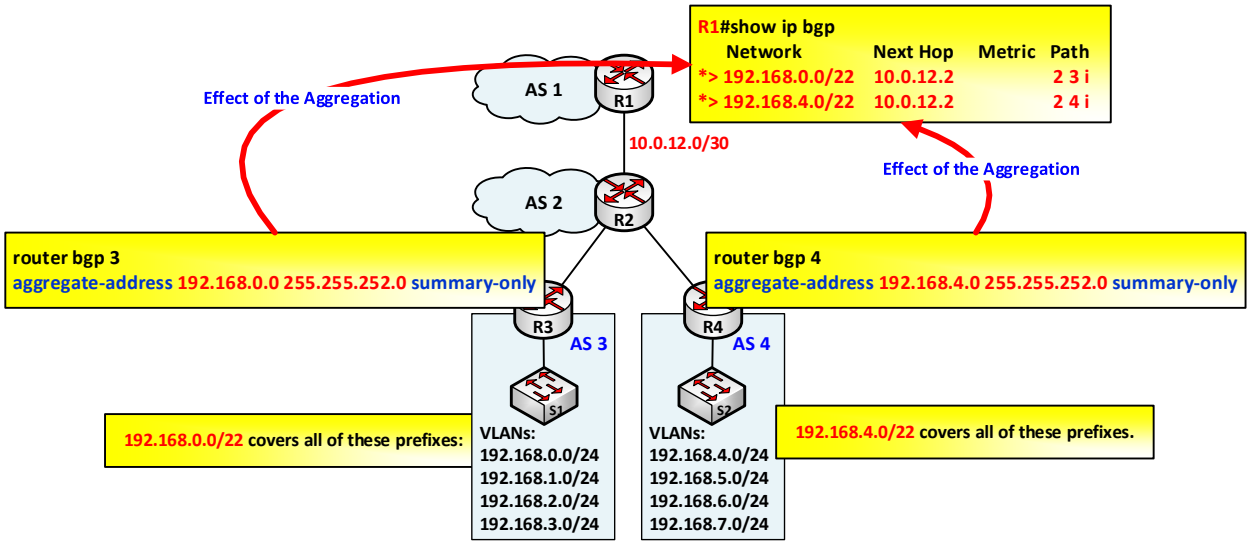


Figure 8.44

The IP prefix 192.168.4.0/22 aggregates the block of addresses starting in 192.168.4.0/24 and ending in 192.168.7.0/24 as this next binary tree shows.

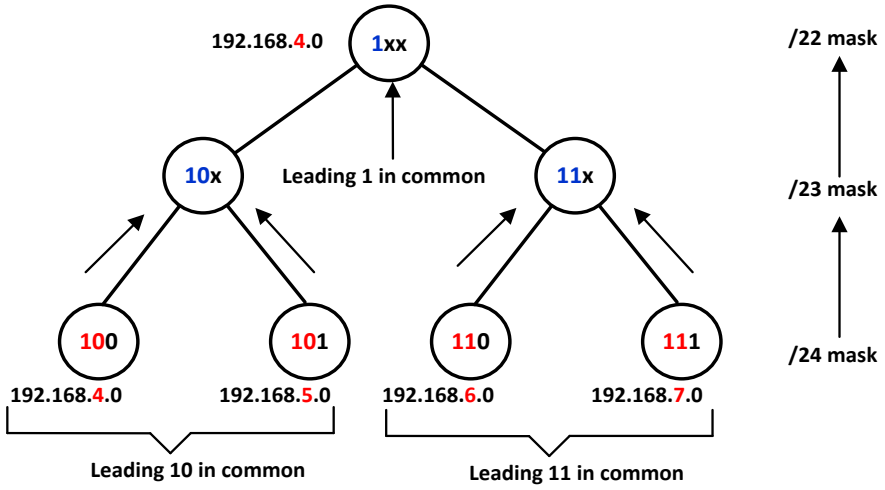


Figure 8.45

Juniper Implementation

Let's add the right side of the network to show how the aggregation is implemented in JUNOS. Only the configuration of J3 is explained in detail since the configuration of J4 is similar.

First, the aggregate routes must be created. In J3 the aggregate route 172.16.0.0/17 covers all the prefixes in AS30 while the policy to-AS20 controls the exporting of information to J2 in AS20.

```
set routing-options aggregate route 172.16.0.0/17
```

Once created the aggregation, a statement with a route-filter matches the supernet exactly.

```
set policy-options policy-statement to-AS20 term 1 from protocol aggregate
set policy-options policy-statement to-AS20 term 1 from route-filter 172.16.0.0/17 exact
set policy-options policy-statement to-AS20 term 1 then accept
```

Finally, a statement in the BGP group will export the routing information to J2.

```
set protocols bgp group EXTERNAL export to-AS20
```

All together:

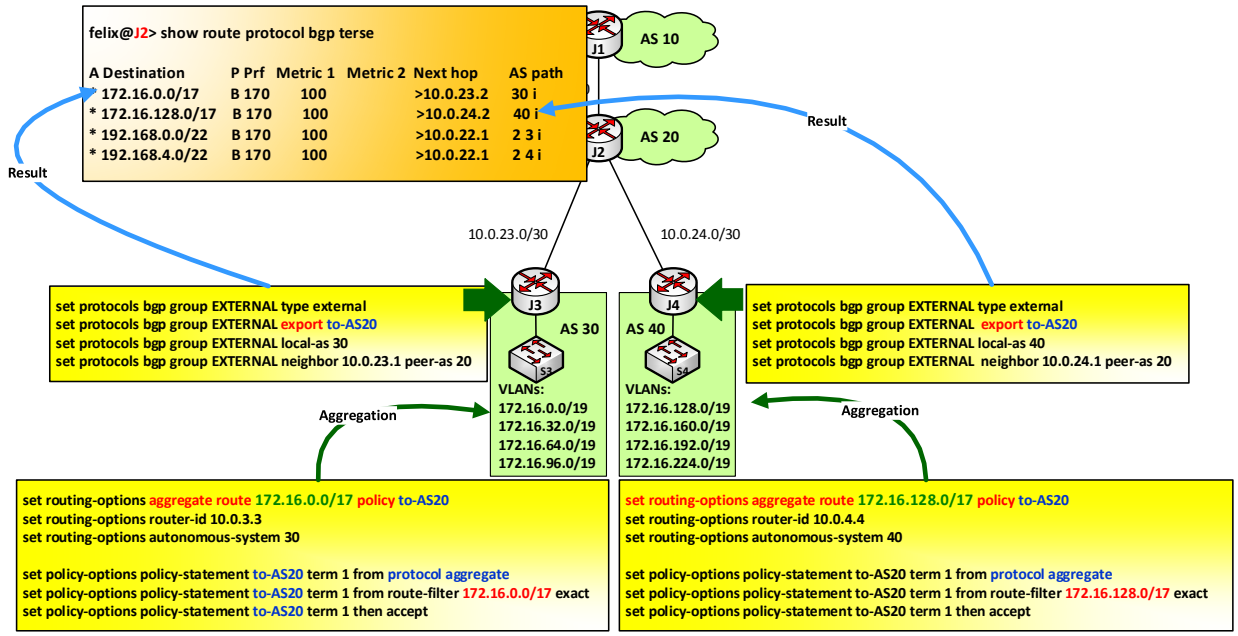


Figure 8.46

The results of the aggregation are observed in J2 where the number of BGP advertised entries has decreased from 16 entries to only four supernet entries. The aggregations 192.168.0.0/22 and 192.168.4.0/22 are coming from the previously configured Cisco routers R3 in AS3 and R4 in AS4.

Aggregation at the Provider Level

There are situations when the customer does not implement the aggregation but rather the service provider. Let's explore this case here. Router R2 will perform the aggregation instead of the customer routers R3 and R4. The idea is that R1 will get very compacted routing information about the lower network tiers. In router R2, the following commands are issued:

```
R2(config)#router bgp 2
R2(config-router)#aggregate-address 192.168.0.0 255.255.252.0 summary-only
R2(config-router)#aggregate-address 192.168.4.0 255.255.252.0 summary-only
```

This is the result in R1:

```
R1#show ip bgp
  Network          Next Hop        Metric LocPrf Weight Path
*> 192.168.0.0/22  10.0.12.2         0             0    2 i
*> 192.168.4.0/22  10.0.12.2         0             0    2 i
```

In R1, only two supernets appear in the BGP database but notice that the information about the sourcing autonomous systems (AS3 and AS4) has been lost. Technically, the aggregations are created in AS2, but it would be more accurate if the AS-PATH preserved the actual origin AS information. Consequently, Cisco IOS has the keyword "**as-set**" to do that:

```
R2(config)#router bgp 2
R2(config-router)#aggregate-address 192.168.0.0 255.255.252.0 as-set summary-only
R2(config-router)#aggregate-address 192.168.4.0 255.255.252.0 as-set summary-only
```

This is the result now in R1:

```
R1#show ip bgp
  Network          Next Hop        Metric LocPrf Weight  Path
*> 192.168.0.0/22  10.0.12.2         0             0    2 3 i
*> 192.168.4.0/22  10.0.12.2         0             0    2 4 i
```

The actual information about the path is present now. One would think that the "as-set" subcommand should be acting by default, but there is a case where not including the information about the source AS is acceptable. In effect, an Internet Service Provider could be leasing address space to its customers which would have private autonomous systems numbers. In this situation, the removal of the initial AS numbers makes sense.

To further illustrate the effect of the "as-set" sub-command, let's assume that the ISP decides to aggregate all the prefixes from AS3 and AS4 under just one supernet. When 192.168.0.0/21 covers the two blocks of addresses, the AS-PATH information about the autonomous systems (AS3 and AS4) that originated the routing entries becomes ambiguous. What is the source of the AS-PATH now? Is it AS3 or is it AS4 or both or none? In this particular situation, the sub-command "**as-set**" have another function.

```
R2(config)#router bgp 2
R2(config-router)#aggregate-address 192.168.0.0 255.255.248.0 as-set summary-only
```

This is the result in R1:

```
R1#show ip bgp
  Network          Next Hop        Metric LocPrf Weight    Path
*> 192.168.0.0/21   10.0.12.2          0              0      2 {3, 4} i
```

The AS-PATH detail {3,4} is a piece of information that indicates to another routing domain, such as AS1, that an aggregation was done at AS2 and it covers prefixes from two different autonomous system AS3 and AS4. This AS-SET information is transmitted inside the field of the AS-PATH Attribute in the BGP UPDATE messages exchanged by the routers. The details of this AS-SET will be explained in a later section by analyzing sniffed BGP data.

Aggregation by Policy at the Provider Level

This is the same model as before but in this case, the Internet Service Provider makes different aggregations depending on the receiving neighbour needs. No aggregation is being done by the customer so the administration in AS2 have agreed to advertise one aggregation block to the upper carrier and two more specific blocks to the ISP in AS20. The advertisements of the aggregations will be controlled with routing policies.

First, let's decide the aggregation of the routes under R2-AS2. Since this router is the pivot to reach AS3 and AS4, it makes sense that it only advertises aggregations to the upper tier routers. This is R2's BGP current database.

```
R2#show ip bgp
  Network          Next Hop    Metric LocPrf Weight  Path
*> 172.16.0.0/19   10.0.22.2                    0    20 30 i
*                  10.0.12.1                    0    1 10 20 30 i
*> 172.16.32.0/19  10.0.22.2                    0    20 30 i
*                  10.0.12.1                    0    1 10 20 30 i
*> 172.16.64.0/19  10.0.22.2                    0    20 30 i
*                  10.0.12.1                    0    1 10 20 30 i
*> 172.16.96.0/19  10.0.22.2                    0    20 30 i
*                  10.0.12.1                    0    1 10 20 30 i
*> 172.16.128.0/19 10.0.22.2                    0    20 40 i
*                  10.0.12.1                    0    1 10 20 40 i
*> 172.16.160.0/19 10.0.22.2                    0    20 40 i
*                  10.0.12.1                    0    1 10 20 40 i
*> 172.16.192.0/19 10.0.22.2                    0    20 40 i
*                  10.0.12.1                    0    1 10 20 40 i
*> 172.16.224.0/19 10.0.22.2                    0    20 40 i
*                  10.0.12.1                    0    1 10 20 40 i
*> 192.168.0.0     10.0.23.2        0           0    3 i
*> 192.168.1.0     10.0.23.2        0           0    3 i
*> 192.168.2.0     10.0.23.2        0           0    3 i
*> 192.168.3.0     10.0.23.2        0           0    3 i
*> 192.168.4.0     10.0.24.2        0           0    4 i
*> 192.168.5.0     10.0.24.2        0           0    4 i
*> 192.168.6.0     10.0.24.2        0           0    4 i
*> 192.168.7.0     10.0.24.2        0           0    4 i
```

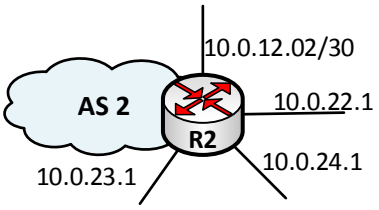


Figure 8.47

This router R2 will summarize three blocks of addresses:

1. The first block will cover the range from 192.168.0.0/24 to 192.168.3.0/24.
2. The second block will cover the range from 192.168.4.0/24 to 192.168.7.0/24.
3. The third block will cover all the range from 192.168.0.0/24 to 192.168.7.0/24.

Three supernets are created with the "aggregate-address" command under BGP.

- The supernet 192.168.0.0 255.255.252.0 (or /22) covers the four IP prefixes from AS3.
- The supernet 192.168.4.0 255.255.252.0 (or /22) covers the four IP prefixes from AS4.
- The supernet 192.168.0.0 255.255.248.0 (or /21) covers the previous two supernets.

This is the implementation in Cisco IOS:

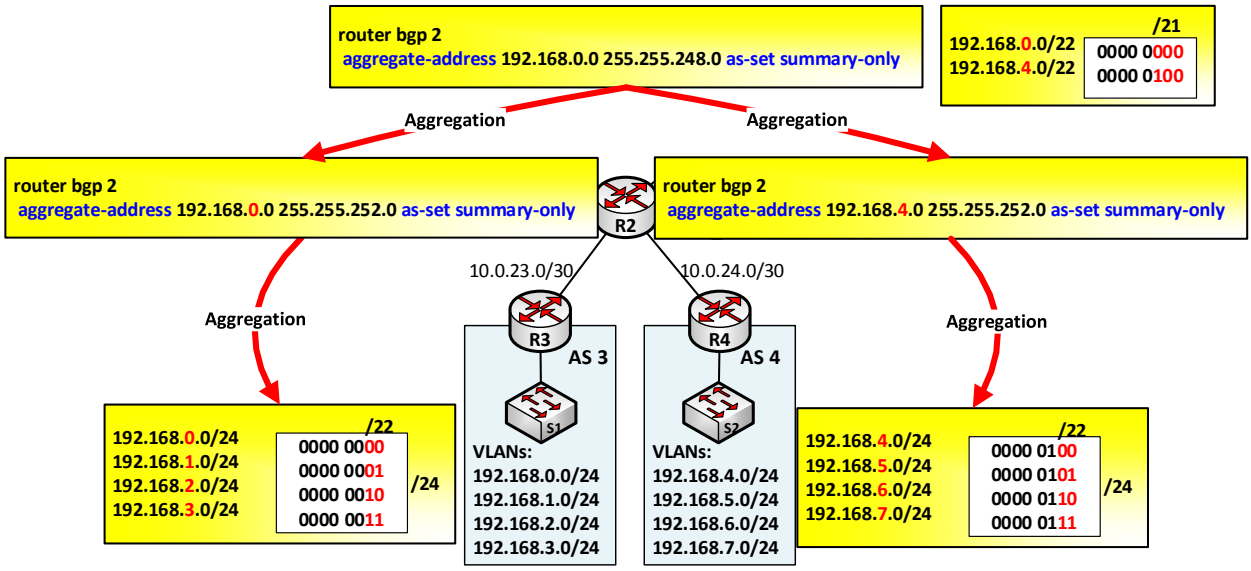


Figure 8.48

Additionally to the aggregation commands a BGP routing policy is needed to control the level of information send out to the neighbours. This is how aggregations will be advertised:

- The first and second block (the /22s) will be advertised to J2-AS20 only.
- One aggregation (/21) covering all the blocks will be advertised to R1-AS1 only.

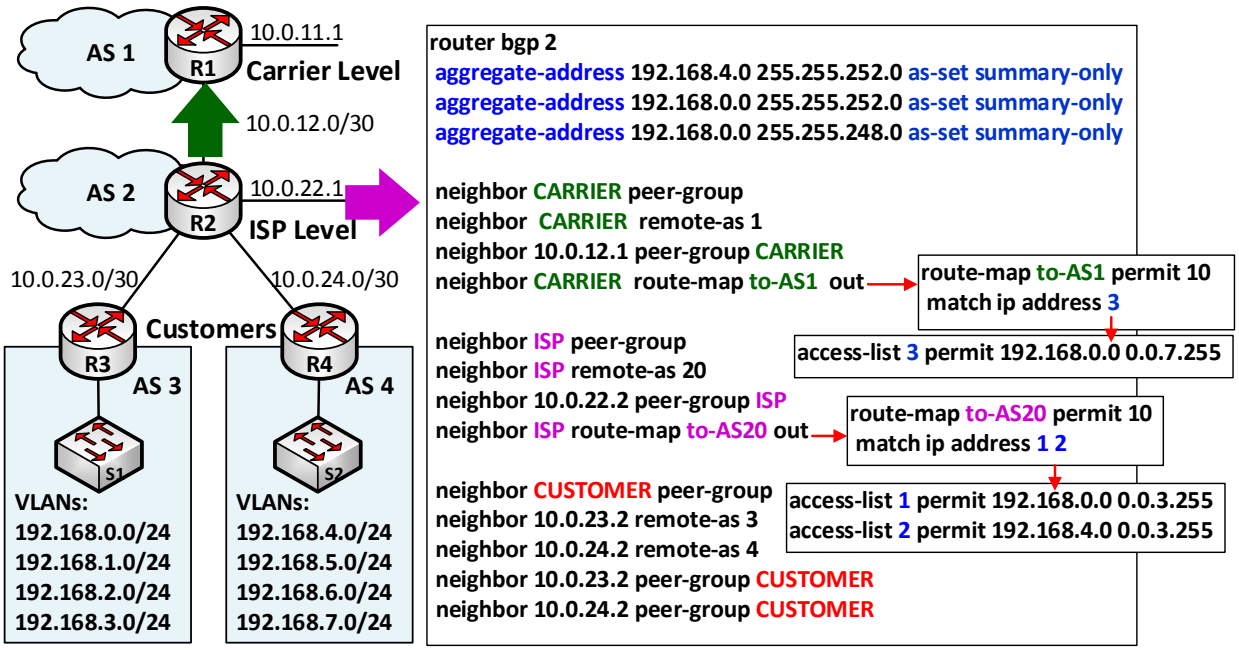


Figure 8.49

R2 uses the policy to-AS1 to control that only 192.168.0.0/21 is sent out to R1. On the other hand, the policy to-AS20 controls that J2 receives 192.168.0.0/22 and 192.168.4.0/22.

The result in R1 validates that the policy is working as intended. Router R1 only knows the aggregation 192.168.0.0/21 that covers all the eight IP prefixes sourced by AS3 and AS4. The AS-PATH shows that the advertisement comes via AS2 and then from either AS3 or AS4. This is indicated with the AS-SET {3, 4}.

```
R1#show ip bgp (* edited)
 Network          Next Hop     Metric LocPrf  Weight    Path
*> 192.168.0.0/21   10.0.12.2     0              0       2 {3, 4} i
(* Plus all the routes from the space 172.16.0.0 not yet aggregated)
```

Router J2 in AS20 should be receiving 192.168.0.0/22 and 192.168.4.0/22. This router also receives the detailed information from the customers AS30 and AS40. Let's observe the effect of this aggregation.

```
felix@J2> show route protocol bgp terse
A Destination        P Prf  Metric 1  Metric 2  Next hop     AS path
* 172.16.0.0/19      B 170    100               >10.0.23.2   30 i
* 172.16.32.0/19     B 170    100               >10.0.23.2   30 i
* 172.16.64.0/19     B 170    100               >10.0.23.2   30 i
* 172.16.96.0/19     B 170    100               >10.0.23.2   30 i
* 172.16.128.0/19    B 170    100               >10.0.24.2   40 i
* 172.16.160.0/19    B 170    100               >10.0.24.2   40 i
* 172.16.192.0/19    B 170    100               >10.0.24.2   40 i
* 172.16.224.0/19    B 170    100               >10.0.24.2   40 i
* 192.168.0.0/21     B 170    100               >10.0.12.1   10 1 2 {3 4} i
* 192.168.0.0/22     B 170    100       0       >10.0.22.1   2 3 i
* 192.168.4.0/22     B 170    100       0       >10.0.22.1   2 4 i
```

The size of the table has been reduced and now J2 has three aggregations as intended by the policy in R2. It is important to point out that the router will keep the three entries in the routing table even though the /21 covers the address space of the other two aggregations because fundamentally they are different route prefixes. Furthermore, the IP protocol will always prefer the more specific /22 prefixes (IP longest match) to reach destinations. However, in case that

the link between R2 and J2 (10.0.22.0/30) were to fail; there is an alternative path to the destinations in AS3 and AS4. This is thanks to the aggregation 192.168.0.0/21 which is advertised via J1. So far, the implementation seems to work fine.

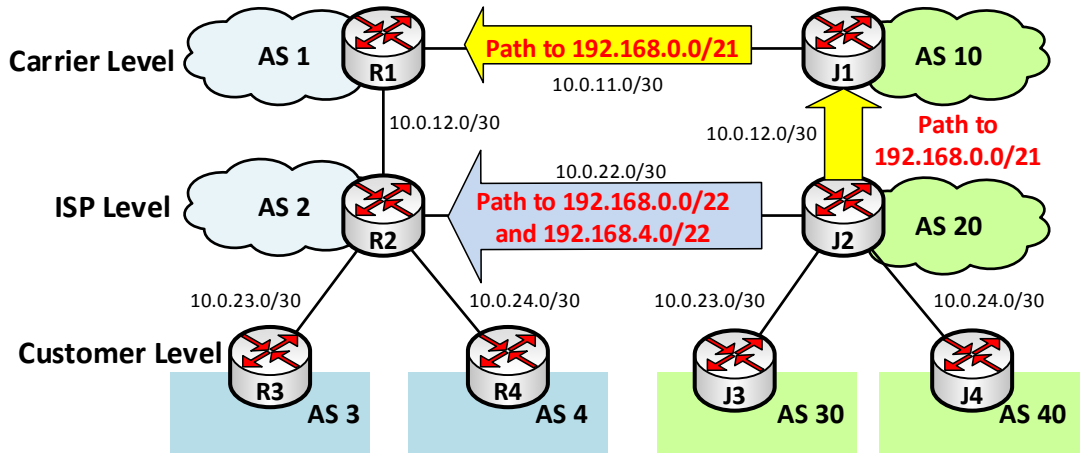


Figure 8.50

It is the turn to aggregate the prefixes under the Juniper routers in AS30 and AS40. The following configuration is done in router J2.

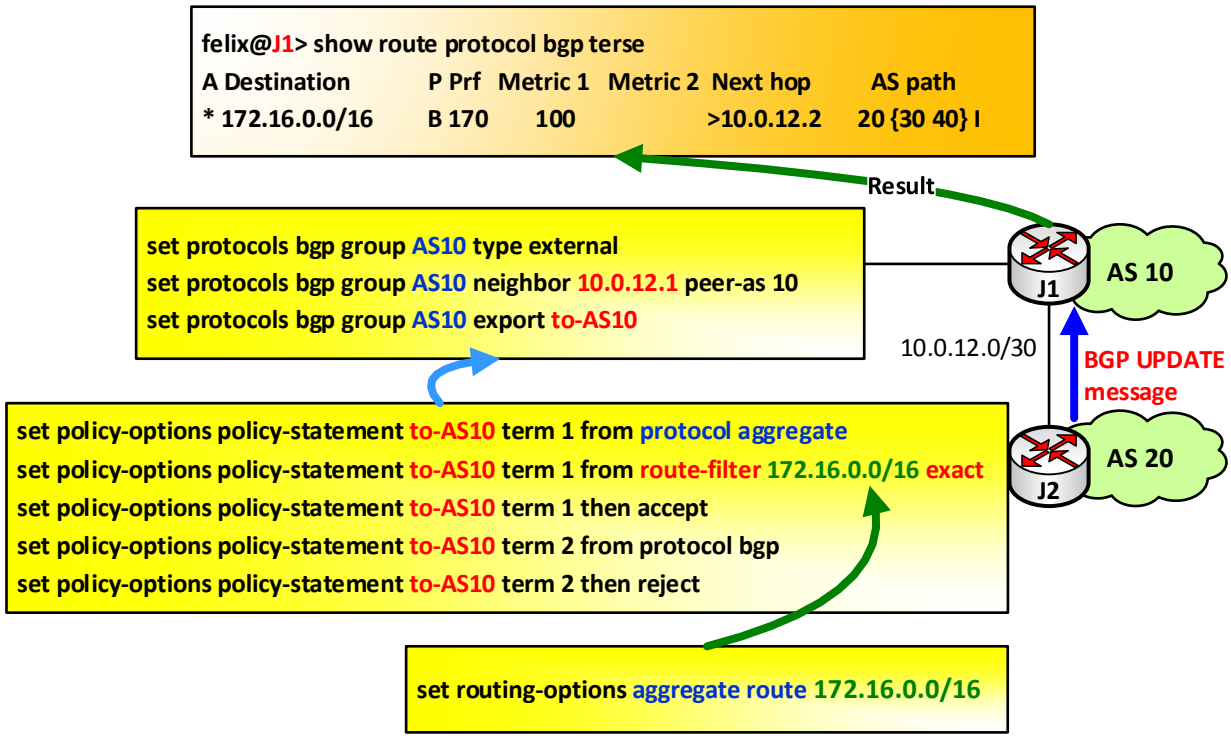


Figure 8.51

First, the aggregate route 172.16/16 is to be advertised to J1. This aggregation contains all the address space underneath AS30 and AS40 contiguously. In JUNOS, the aggregation can be matched from either the routing table entries or in this case, from the "protocol" aggregate. (This is a JUNOS concept; the protocol does not really exist). A route filter makes sure that the prefix is matched exactly. Term 2 matches other routes advertised by protocol BGP and then it rejects them. This is actually filtering out the more specific prefixes from AS30 and AS40 and anything else. Finally, the result of the policy to-AS10 is exported toward J1. The result of the successful aggregation can be seen in J1.

The second part of the configuration is for advertising the more specific aggregations 172.16.0.0/17 and 172.16.128.0/17 to R2. The policy to-AS2 is directed toward router R2-AS2.

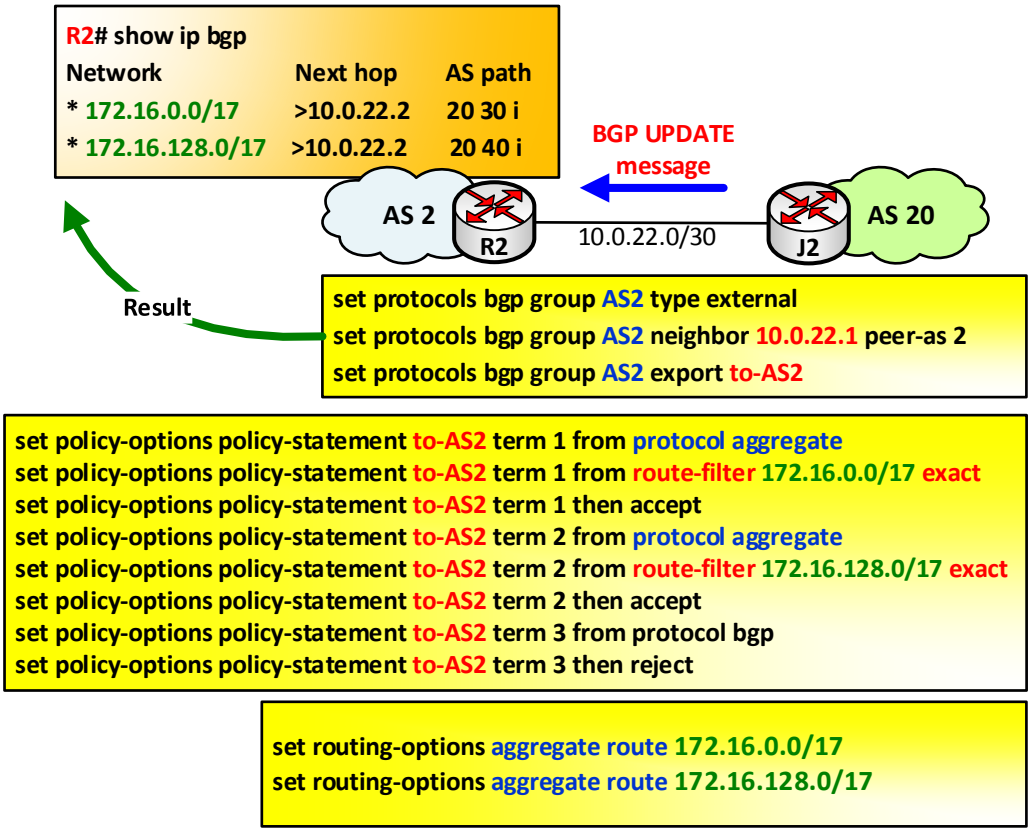


Figure 8.52

The following diagram puts all the routing information together. Miscellaneous information have been omitted to allow a more clear view. The two top routers R1 and J1 know only the two general aggregations 172.16/16 and 192.168.0.0/21 while the routers R2 and J2 know the previous aggregations plus the more detailed aggregations. If link 10.22.22.0/30 failed, the general aggregations would point to an alternative path via the top routers R1 and J1.

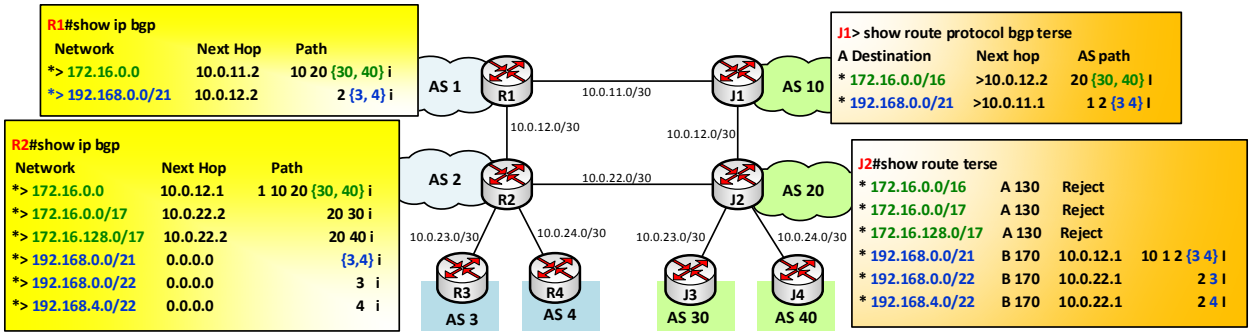


Figure 8.53

The scenario have been completed, there is one more concept to test yet. Since the aggregations are created manually by the administrator, what happens if a block of addresses is withdrawn? Would the aggregations be withdrawn too? Let's test this concept by shutting down completely routers J3 and J4. In principle, all the 172.16 aggregations should disappear too because a routing protocol should not advertise a black hole. This is the routing table in router J2 after shutting down J3 and J4.

```
felix@J2> show route terse
A Destination
* 192.168.0.0/21
* 192.168.0.0/22
*192.168.4.0/22
* 10.0.2.2/32
* 10.12.12.0/30
* 10.22.22.0/30
* 10.23.23.0/30
* 10.24.24.0/30
```

In effect, the aggregations from the 172.16 space have disappeared as expected. Router J3 is being rebooted right now and this happens in J2:

```
* 10.0.2.2/32        D  0
* 10.0.12.0/30     D  0                    >em0.0
* 10.0.22.0/30     D  0                    >em4.0
* 10.0.23.0/30     D  0                    >em3.0
* 10.0.24.0/30     D  0                    >em1.0
* 172.16.0.0/16    A 130                     Reject
* 172.16.0.0/17    A 130                     Reject
* 172.16.0.0/19     B 170      100         >10.0.23.2     30 i
* 172.16.32.0/19   B 170      100          >10.0.23.2     30 i
* 172.16.64.0/19   B 170      100          >10.0.23.2     30 i
* 172.16.96.0/19   B 170      100          >10.0.23.2     30 i
```

As soon as J3 advertises its prefixes to J2 that triggers the activation of the corresponding aggregation 172.16.0.0/17 and that is all right. However, the more general 172.16.0.0/16 also reappears and this one covers the address space under AS40 which is out of service right now. Let's observe the effect in J1 and R2 (data edited for simplicity):

```
felix@J1> show route terse

A Destination        Next hop        AS path

* 172.16.0.0/16      >10.12.12.2       20 30 i

* 192.168.0.0/21     >10.11.11.1       1 2 {3 4} i
* 10.0.1.1/32
* 10.0.11.0/30
* 10.0.12.0/30
```

```
R2#show ip bgp

  Network            Next Hop     Path

*>172.16.0.0         10.0.12.1    1 10 20 {30, 40} i

* >172.16.0.0/17     10.0.22.2                20 30 i
```

The general aggregation gets installed in both J1 and R2. That means that if traffic were directed toward one of the four networks located inside AS40, it will be eventually delivered to J2 where it seems to be a path to the destination. Nevertheless, J2 will not have the entries pointing to the specific four networks so the traffic will be discarded right there. In the current setting that is something that might happen and that teaches a lesson which is that aggregations can cause strange problems difficult to troubleshoot in large, complex network topologies. Consequently, they must be implemented carefully.

In conclusion, aggregations are enabled if there is at least one route active in the block being aggregated. As a general rule, to avoid advertising black holes, aggregations should be done as close as possible to the source of the traffic. Ideally, routing domains at the edges of the Internet should be doing their aggregations. Organizations, such as the CIDR Report [1], continuously publish aggregation recommendations.

Aggregation and BGP Attributes

When aggregation is implemented, the BGP UPDATES will carry additional information that helps network administrators to understand where and how the aggregation was done. Aggregation triggers the activation of two BGP attributes; the **Atomic Aggregate** and the **Aggregator**. Also the **AS PATH** attribute might be affected by the aggregations depending of the options used for the configuration of the aggregations.

AS Path Attribute

The AS-PATH attribute is well-known so all routers must implement and understand such attribute as it is a fundamental for the functioning of BGP. The attribute is also transitive as it is passed along from an autonomous system to the next. The attribute is considered by the BGP decision making algorithm in the steps for the determination of a best path to a destination, so it is critical that it contains accurate information.

The **AS PATH** attribute is structured in two types of **segments**, one is the **AS_SEQUENCE** and the other is the **AS_SET** segment. The AS_SEQUENCE segment lists the autonomous system numbers in the path to the destination route from the closest one to the originator. Paths without aggregations are described by default by this type of segment. The AS_SET segment only appears when aggregation is implemented with the AS_SET option for covering the address space of two or more autonomous systems.

Continuing with the previous network topology, a BGP UPDATE has been sniffed and its relevant contents are exposed in the next diagram. The UPDATE is sent by R1 to R2 and it shows:

- the AS_PATH with the **AS-SEQUENCE** of AS1, AS10, AS20
- and then the aggregation with an **AS_SET** of AS30 and AS40.

Border Gateway Protocol
Marker FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
Length 75 bytes
TYPE UPDATE message (2) (from R1 to R2)
Unfeasible routes length 0 bytes
Total path attributes length 49 bytes
Path attributes:
ORIGIN: IGP
AS_PATH: 1 10 20 {30, 40}
Flags: well-known, transitive, complete
Type code: AS_PATH (2)
Length: 24 bytes
AS path segment: 1 10 20
Path segment type: AS_SEQUENCE (2)
Path segment length: 3 AS
Path segment value: 1 10 20
AS path segment: {30, 40}
Path segment type: AS_SET (1)
Path segment length: 2 AS
Path segment value: 30 40

NEXT_HOP: 10.0.12.1
AGGREGATOR: AS: 20 origin 10.0.2.2
(this is J2@AS20)
Network Layer Reachability Information NLRI: 3B

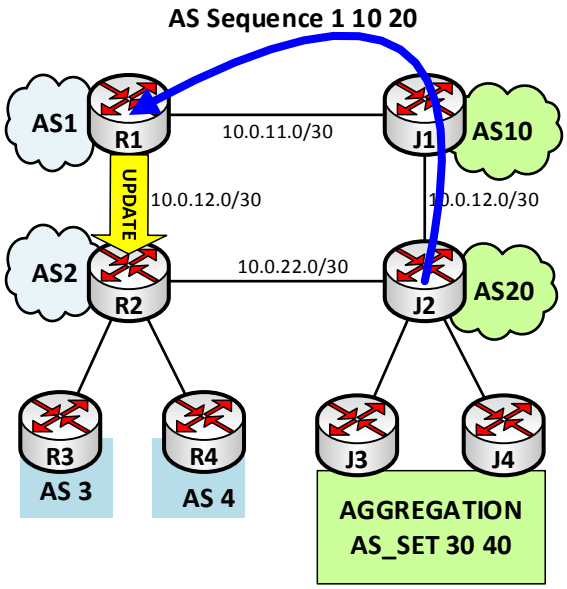


Figure 8.60

This AS_PATH contains all the information that a network engineer will need to understand the path followed to reach the aggregation.

Atomic Aggregate Attribute

The Atomic Aggregator is an alert to indicate that an aggregation has happened. BGP attribute number 6 is well-known so all BGP routers must understand and it is transitive so it must be passed along from AS to AS.

This attribute can only be seen inside the BGP UPDATE when an aggregation is done but without an AS_SET. For instance, when network aggregation is performed by border router R3 for the prefixes in AS3 then there is no point in setting the AS_SET because it is only one AS. In this case, the Atomic Aggregate Attribute gets added to the attribute lists in the BGP UPDATE that goes from R3 to R2 and then it is passed along to R1.

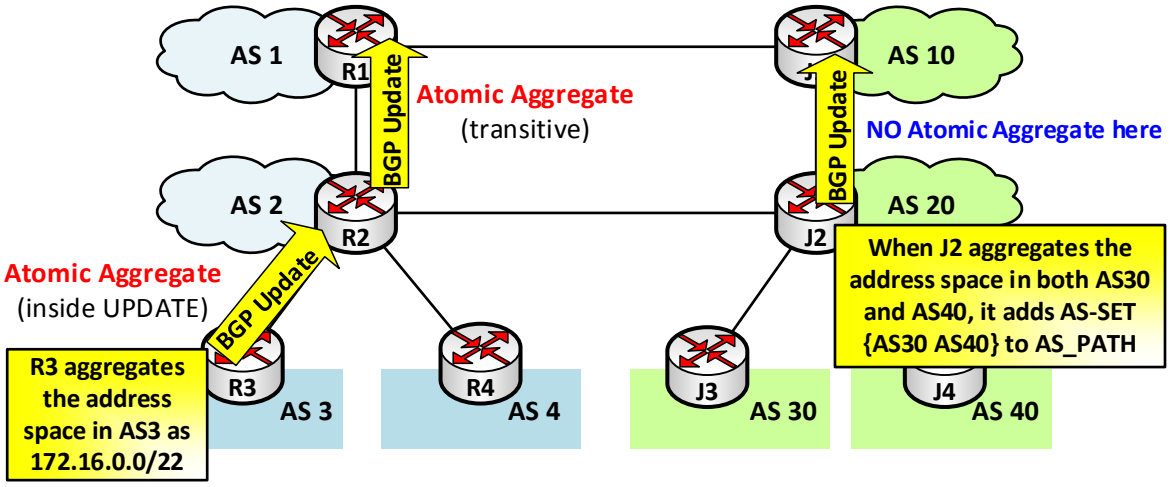


Figure 8.61

On the other hand, the Atomic Aggregate Attribute is not present when the AS_SET is activated. For example, if the address spaces of AS30 and AS40 are aggregated with the AS_SET option, then the Atomic Aggregate Attribute is not included in the BGP UPDATE because the more specific information makes it irrelevant.

Aggregator Attribute

The Aggregator Attribute tells the identification of the router that did the aggregation and the autonomous system where such router is located. In effect, the following snippet shows that the aggregation prefix 192.168.0.0/21 was originated by router R2 with the identification 10.2.2.2. In this case, the router-id is the loopback's IP address.

Border Gateway Protocol

Marker FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

Length 78 bytes

TYPE: UPDATE message (2) (from R2 to R1)

Unfeasible routes length 0 bytes

Total path attributes length 51 bytes

Path attributes:

ORIGIN: IGP

AS_PATH: 2 {3, 4}

NEXT_HOP: 10.0.12.2

MED: 0

ATOMIC_AGGREGATE: 3 bytes

AGGREGATOR: AS: 2 origin 10.2.2.2 (this is R2@AS2)

Network Layer Reachability Information NLRI:

192.168.0.0/21

Figure 8.62

The Aggregator is the attribute Type 7 and it is optional, transitive and complete. The attribute only contains the two fields described above, the number of the origin autonomous system and the identification address of the router that created the aggregation.

The following network case has been designed to showcase the effects of the "summary-only" and "as-set" Cisco sub-commands in the attributes related to aggregation carried by the BGP UPDATE message. Three attributes are examined here: the AS_PATH, the ATOMIC_AGGREGATE, and the AGGREGATOR. The BGP messages were sniffed for the left side in 10.0.12.1 and for the right side in 10.0.13.1.

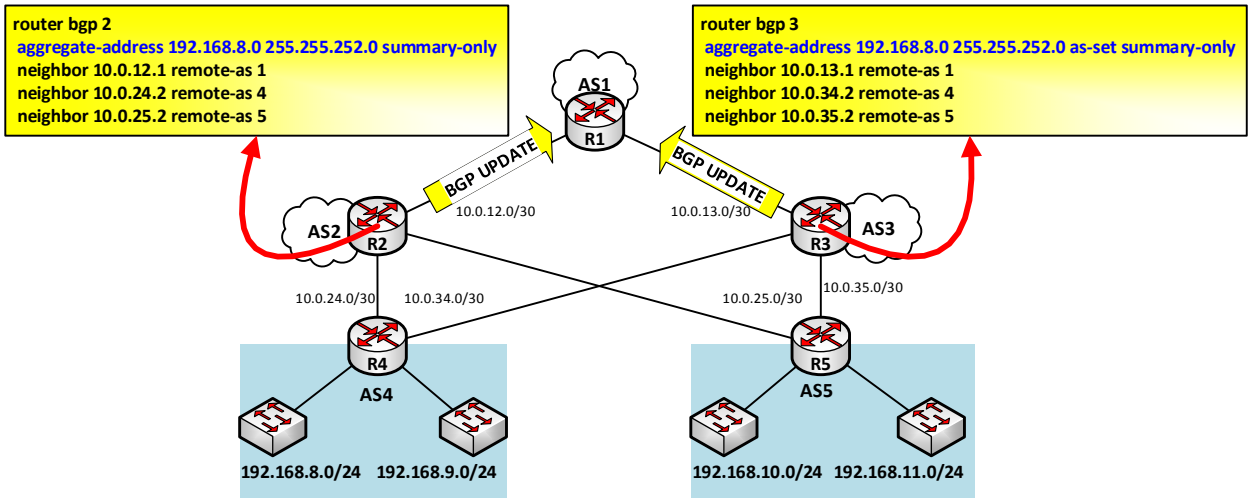
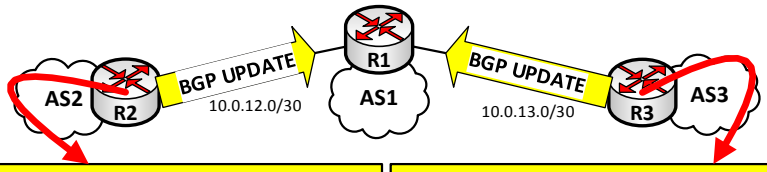


Figure 8.63

Router R2, in the left side of the network, aggregates the prefixes from AS4 and AS5 with the supernet 192.168.8.0/22 using the "summary-only" sub-command. The AS_PATH attribute only carries an AS_SEQUENCE segment of AS2 while the details of AS4 and AS5 are lost in the aggregation. In short, AS2 appears to contain all the IP prefixes. The ATOMIC_AGGREGATE Attribute is included to emphasize that an aggregation has occurred and the AGGREGATOR Attribute specifies that the router with the identification 10.2.2.2 in AS2 did the aggregation.




```
router bgp 2
 aggregate-address 192.168.8.0 255.255.252.0 summary-only
```

```
router bgp 3
 aggregate-address 192.168.8.0 255.255.252.0 as-set summary-only
```

Border Gateway Protocol
Marker FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
Length 75 bytes
TYPE UPDATE message (2) (from R2 to R1)
Unfeasible routes length 0 bytes
Path attributes: (edited)

AS_PATH: 2
Flags: well-known, transitive, complete
Type code: AS_PATH (2)
Length: 6 bytes
AS path segment: 2
Path segment type: AS_SEQUENCE (2)
Path segment length: 1 AS
Path segment value: 2

ATTOMIC_AGGREGATE: (3 bytes)
Flags: well-known, transitive, complete
Type code: 6

AGGREGATOR: AS: 2 origin 10.2.2.2
Flags: optional, transitive, complete
Type code: 7
Length: 8 bytes
Aggregator AS: 2
Aggregator origin: 10.2.2.2

Network Layer Reachability Information NLRI:
192.168.8.0/22

Border Gateway Protocol
Marker FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
Length 75 bytes
TYPE UPDATE message (2) (from R3 to R1)
Unfeasible routes length 0 bytes
Path attributes (edited)

AS_PATH: 3 {4, 5}
Flags: well-known, transitive, complete
Type code: AS_PATH (2)
Length: 16 bytes
AS path segment: 3
Path segment type: AS_SEQUENCE (2)
Path segment length: 1 AS
Path segment value: 3
AS path segment: {4, 5}
Path segment type: AS_SET (1)
Path segment length: 2 AS
Path segment value: 4 5

AGGREGATOR: AS: 3 origin 10.3.3.3
Flags: optional, transitive, complete
Type code: 7
Length: 8 bytes
Aggregator AS: 3
Aggregator origin: 10.3.3.3

Network Layer Reachability Information NLRI:
192.168.8.0/22

Figure 8.64

On the right side, router R3 in AS3 aggregates 192.168.8.0/22 with the "as-set" sub-command. The BGP message shows that the AS_PATH attribute contains an AS_SEQUENCE of AS3 and then an AS_SET of {4 5} indicating that the aggregation covers address space from two different autonomous system. The AGGREGATOR attribute shows that the aggregation origin is 10.3.3.3 in AS3. Finally, notice that there is no ATOMIC_AGGREGATE because this attribute is suppressed when the AS_SET is activated.

Aggregation to control inbound traffic

This is the typical case of a routing domain that obtains Internet access from several links via one or more service providers (ISP). As an example, let's begin with a very abstract topology.

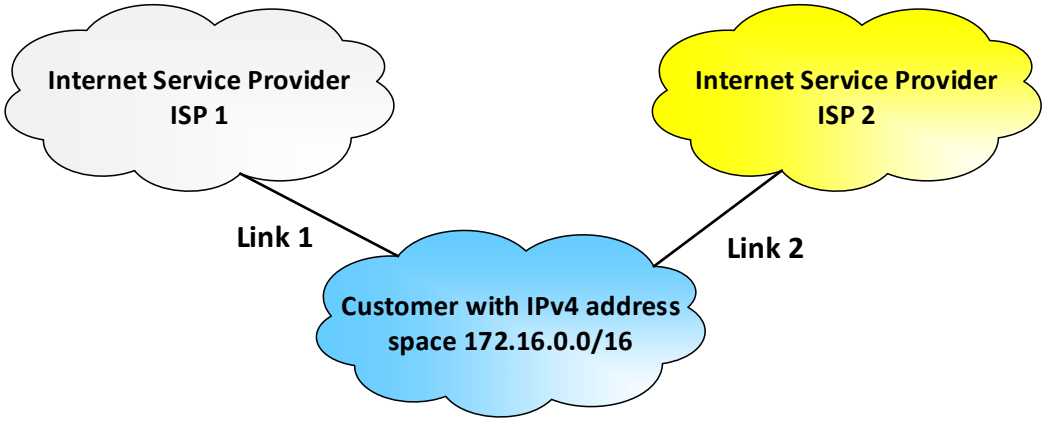


Figure 8.65

In this case, suppose that the customer owns the address space 172.16.0.0/16 and let's take the liberty of assuming that this space is global not private (it is an example after all). Internally, the administration has allocated this IP space to hundreds of networks. Suppose also that there are 254 networks with /24 masks. What should the customer network advertise to the Internet Service Providers?

Certainly, the ISPs would not appreciate getting their routing tables clogged with the hundreds of IP network prefixes. Beside, there is no benefit to advertise all the subnets in this configuration. So the customer should just advertise the aggregation 172.16/16 and it would keep the subnetting details to itself.

But sometimes, advertising more specific information has a purpose. For instance that the customer's data centre is located in the networks 172.16.6.0/24 and 172.16.7.0/24. Then this particular company would like to advertise such prefixes since most of its application services are located there.

Furthermore, the Link1 leading to IPS1 is a very high velocity, expensive connection while the Link2 leading to ISP2 is of lower velocity and cheaper price. The customer would prefer that the traffic directed to the application servers would flow on Link1 while the rest of the more general traffic would flow on Link2. Then some policies for traffic manipulation are needed.

One easy way to implement that would be to advertise the specific prefixes 172.16.6.0/24 and 172.16.7.0/24 over Link1 and then advertise 172.16.0.0/16 over Link2. Since the IP protocol prefers longest matches to reach destinations, the specific prefixes would be more likely reached via Link1 while any other destination traffic would flow via Link2. Additional measures such as matching metrics (MED) or AS-Prepending would complete the scenario.

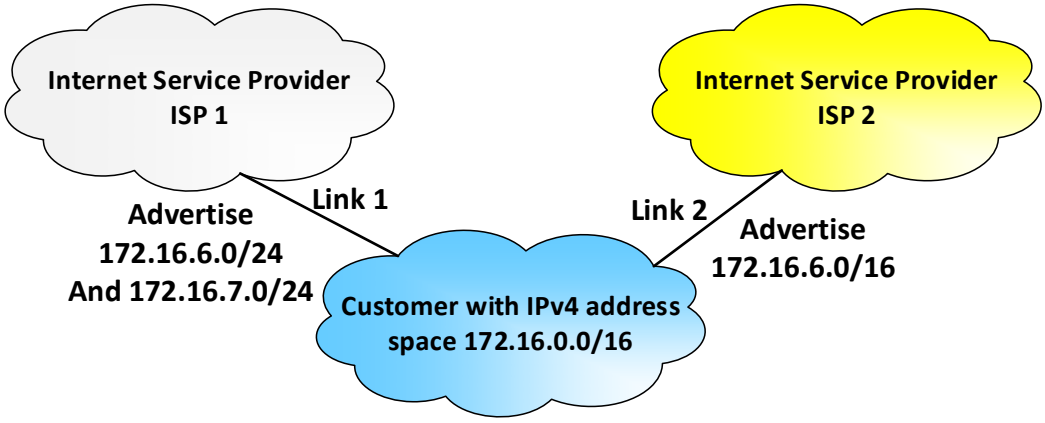


Figure 8.66

When designing a policy, different scenarios must be considered, even improbable ones. If Link1 were to fail, the specific prefixes would still be reachable via Link2 since 172.16.0.0/16 covers them. However, if Link2 fails, only the specific prefixes would be reachable and anything else would not be reachable. A better policy to fix this potential problem would be this:

- Keep advertising 172.16.0.0/16 over Link2.
- Supernet the prefixes 172.16.6.0/24 and 172.16.7.0/24 with 172.16.6.0/23.
- Advertise 172.16.6.0/23 plus 172.16.0.0/16 over Link1, but make 172.16.0.0/16 less desirable by increasing the AS-PATH or the metric depending on the topology.

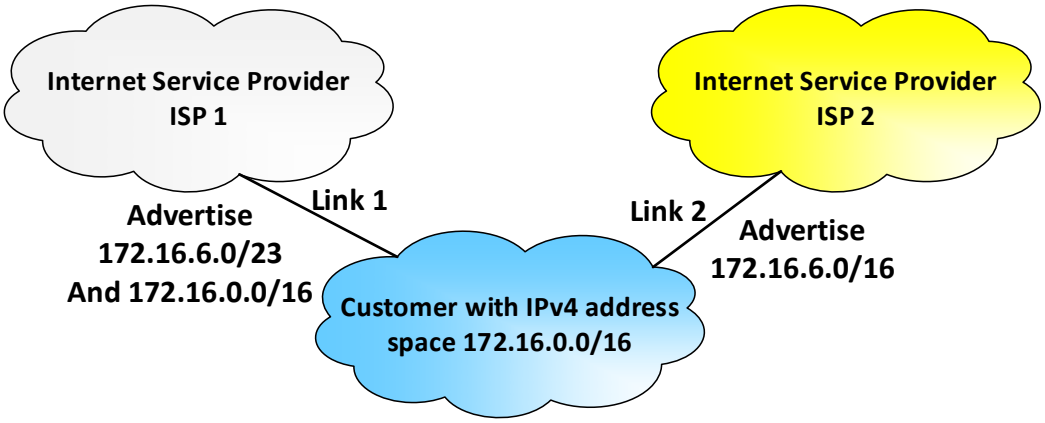


Figure 8.67

In case of any link failure now, all destinations are reachable. Under regular conditions, Link1 is used for traffic with preference while Link2 is used for bulk traffic which is what the customer wants. This would be the detailed topology to do the configuration.

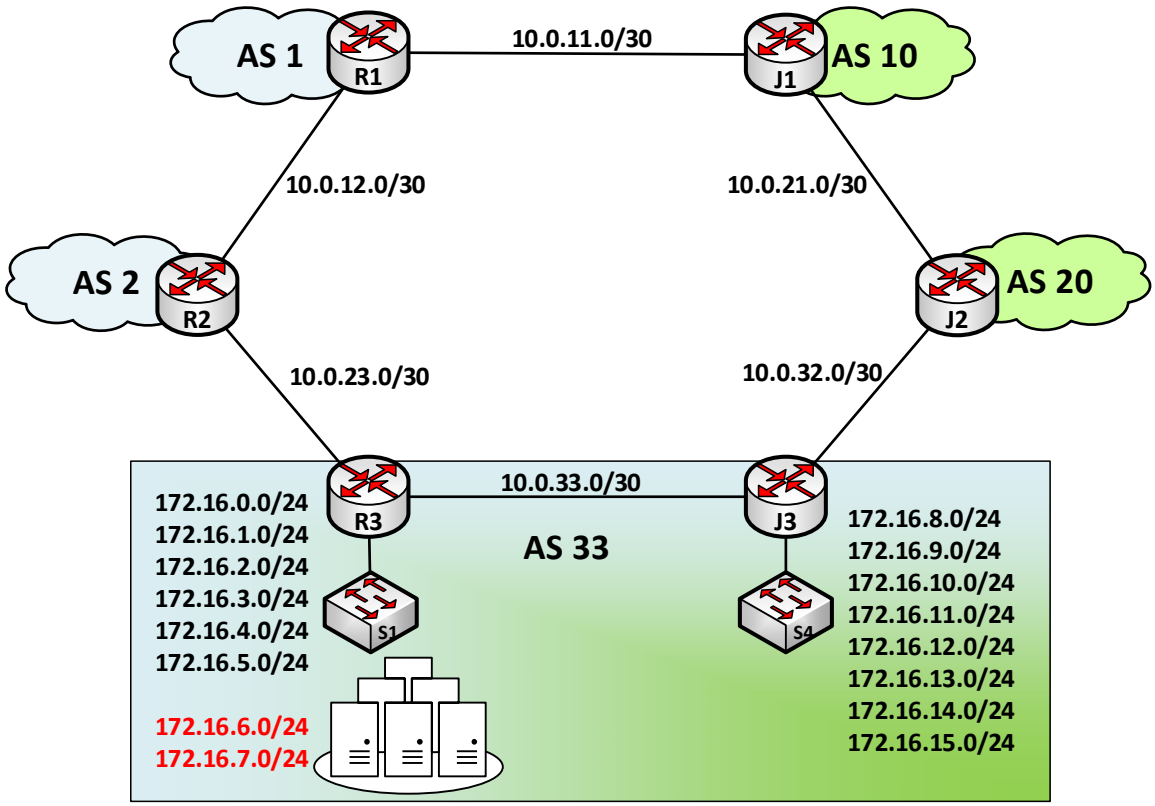


Figure 8.68

For practical reasons, only 16 networks are configured inside AS33 instead of the 254 networks indicated before. The main core of the configuration and its result are summarized in the next diagram.

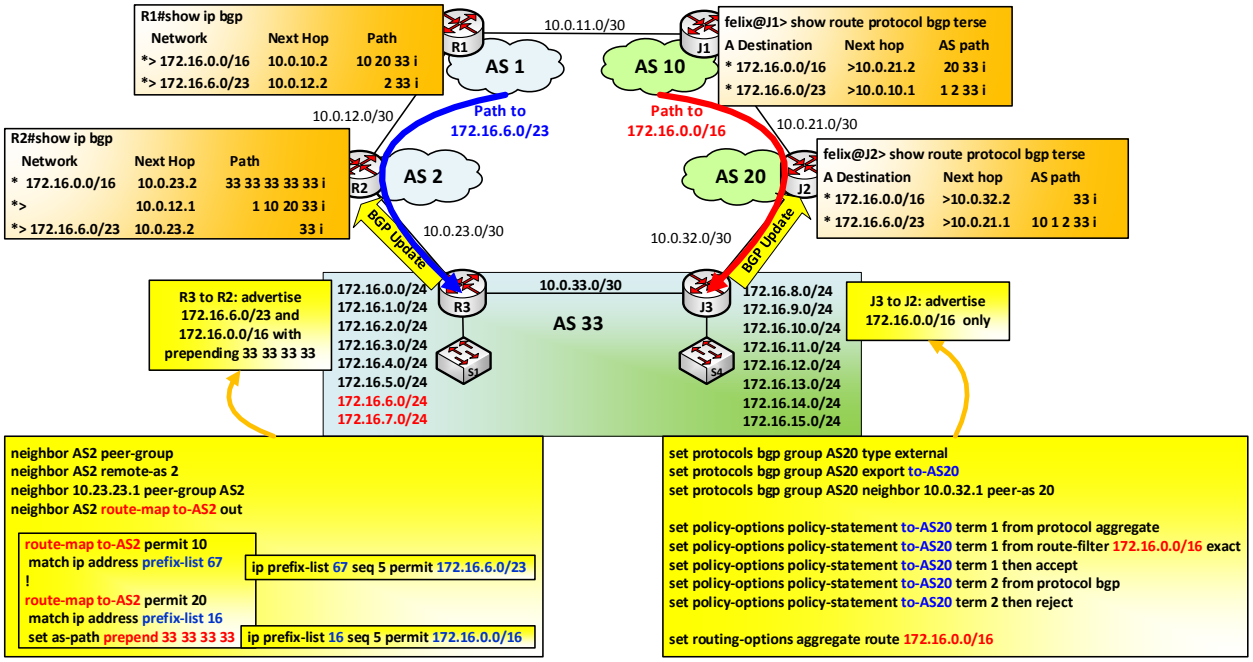


Figure 8.69

Router R2 receives the routing UPDATE from R3 indicating that 172.16.0.0/16 has an AS-PATH of 33 33 33 33 with all the AS prepending done in R3. Router R2 installs that in the BGP database, but there is another path to 172.16.0.0/16 advertised by R1 which has shorter AS-PATH. So, that path gets promoted to the routing table and all the traffic that could arrive to AS2 with such destination will be forwarded to R1. In fact, the only traffic in the link 10.0.23.0/30 will be flowing to the supernet for the data centre or 172.16.6.0/23. This achieves part of the intent of the routing policy so far.

On the other side, router J2 has a very short path to the aggregation route 172.16.0.0/16 via J3. This was also part of the intention of the routing policy. Also, J2 has a path to 172.16.6.0/23 via J1. This router will forward general traffic toward J3 and the specific traffic that have the longest match on 172.16.6.0/23 will be forwarded toward J1.

The top routers R1 and J1 only have two entries in their routing table. Traffic destined to 172.16.0.0/16 goes toward J3 and traffic destined to 172.16.6.0/32 goes toward R3, just as intended by the administration of autonomous system 33.

Aggregation of IPv6 Address Space

The aggregation of address space is a crucial concept in the deployment of IPv6 protocol networks. In fact, the gigantic range of IPv6 addresses could not be handled by any router without supernetting techniques. To place it in perspective, this space has:

- 2^{128} = 340,282,366,920,938,463,463,374,607,431,770,000,000 addresses
- That is over 340 million of millions of millions of millions of millions of millions!

The mere thought of having to do subnetting with such an order of magnitude seems unconceivable. Nevertheless, there are ways to do it. This section describes how to aggregate blocks of IPv6 addresses and how to advertise them under the Border Gateway Protocol.

IPv6 Address Scheme

The Internet Protocol version 6 was designed to follow a hierarchical structure where the five Regional Internet Registries [2] control the top level address assignment. These registries are:

- AFRINIC Africa.
- APNIC Asia/Pacific.
- ARIN for North America
- LACNIC Central and South America
- RIPE NCC Europe/Central Asia/Middle East

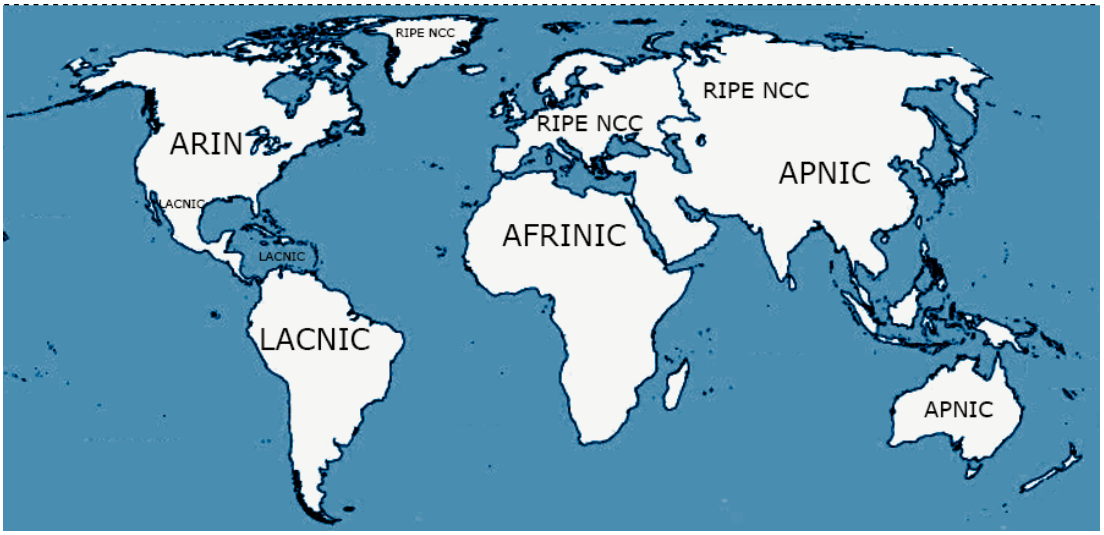


Figure 8.70

It is a common practice that each Regional Registry [4] assigns IPv6 address space in blocks of /24 or /32 masks to carriers and large Internet Service Providers. Then, each provider slices its own address space to give /48 address space to customer sites. Furthermore, each customer organization subnets its own space according to its needs. Graphically:

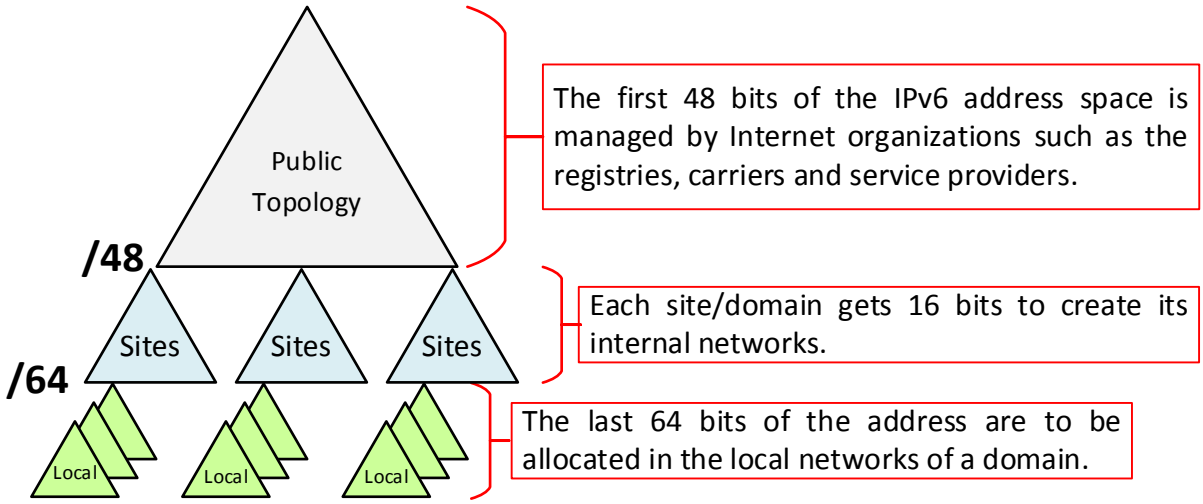


Figure 8.71

Ideally, if this scheme were implemented properly, the Internet top carrier routers would have very few IPv6 prefixes. In a perfect world; whole continents could be represented by the aggregations of the regional registry prefixes. In theory, a border router in Newfoundland could show one aggregation prefix representing North America to another router in Ireland.

In practice, it has not turned out to be quite that way. A query to the Internet global routing table shows that there are close to 40K IPv6 routes on the date that the data was taken. (In any case, that is far less than the 650K prefixes in the IPv4 table.) Observe that, in this edited snippet, the shortest mask length is /32 and the longest is /48. In fact, in the whole table, no mask is longer than /48. In essence, the IPv6 routing table is populated by aggregations.

```
felix@S144> show route terse (edited on July 30, 2017)
inet6: 39,625 destination routes
  2001:: /32
  2001:0004:112:: /48
  2001:0200:: /32
  2001:0200:900:: /40
  2001:0200:8000:: /35
```

IPv6 Address Structure

The prefix 2001:: /32 appears in the previous global routing table. This simple prefix is ideal to explain the structure of an IPv6 address and to introduce a technique to subnet its address space but first, the basic structure of the address needs to be reviewed.

IPv6 addresses [5] are arranged in 8 blocks of 2 bytes each of hexadecimal characters. Then the address 2001:: /32 in expanded format is actually:

2001: 0000: 0000: 0000: 0000: 0000: 0000: 0000 (hexadecimal)

Where 2001: 0000 is prefixed by the mask /32. Let's break this prefix down for further analysis:

$$(20\ 01\ 00\ 00)_{\text{hexadecimal}} = (0010\ 0000\ 0000\ 0001\ 0000\ 0000\ 0000\ 0000)_{\text{binary}}$$

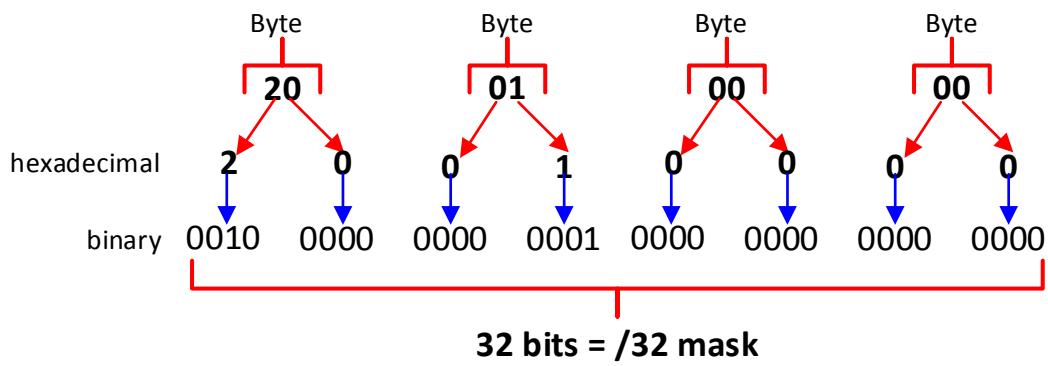


Figure 8.72

- Each hexadecimal coefficient has a binary equivalent of 4 bits.
- Four bits allow to count from $(0000)_{\text{binary}}$ all the way to $(1111)_{\text{binary}}$ which is $(F)_{\text{hexadecimal.}}$
- The hexadecimal coefficient can be any value among 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E or F.

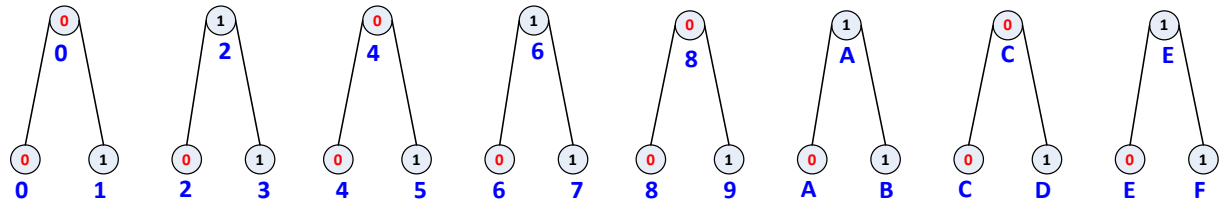
This is a valuable characteristic that will simplify the management of the addresses. The basic knowledge required is to be able to count 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E and F with the four binary bits. Once that the basic structure of the IPv6 address has been outlined, a method to subnet IPv6 address space with a tree can be explained next.

Condition: the following tree system explanation is valid for any $\boldsymbol{mask = n \times 4}$ where n is one of these numbers 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 and 16. That means masks such as /0, /4, /8, /12, /16, /20, /24, /28, /32, /36, /40, /44, /48, /52, /56, /60, /64, etc.

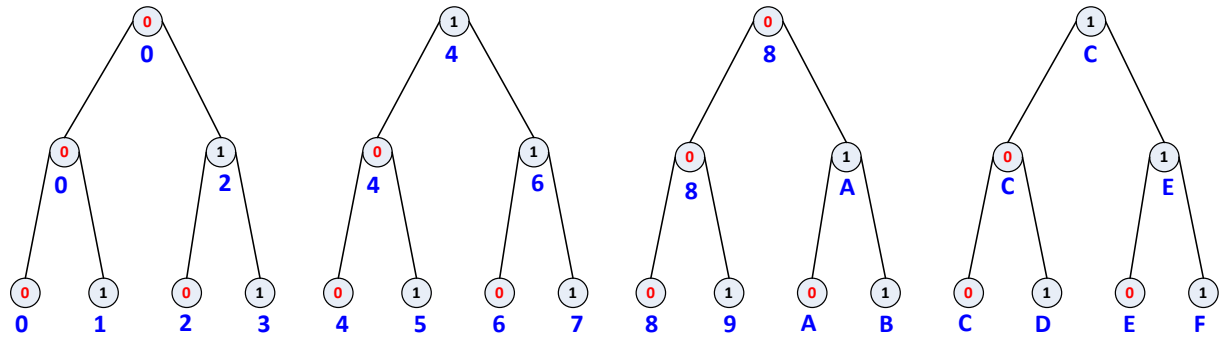
The following graphs outline the IPv6 address allocation method, step by step.

Step 1) Count from 0 to F in hexadecimal. Place a 0 and a 1 alternatively in every circle:

Step 2) Place and link another row of circles on top as it follows:



Step 3) Add another row on top of the previous one like this:



Step 4) Complete the four binary bit tree like this:

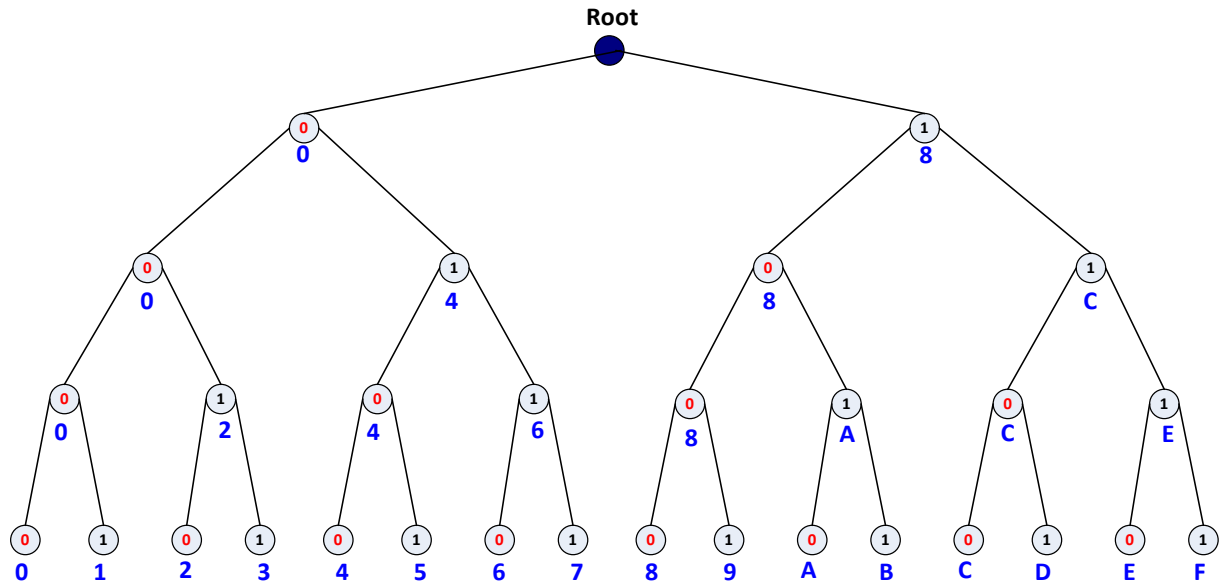


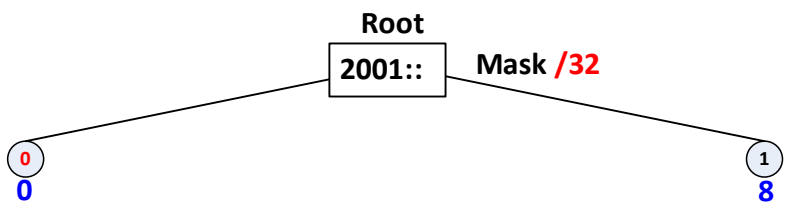
Figure 8.76

Example: the IPv6 prefix 2001: AAAA:: /32 will be used to explain the functioning of the tree.

First) verify that the mask complies with the requirement $(mask = n \times 4)$

- $/32 = 8 \times 4$, so it will work

Second) set the prefix as the root of the tree and place the mask beside it.



Third) keep branching out to complete 5 levels of the tree. Increase the mask by one for every descending level.

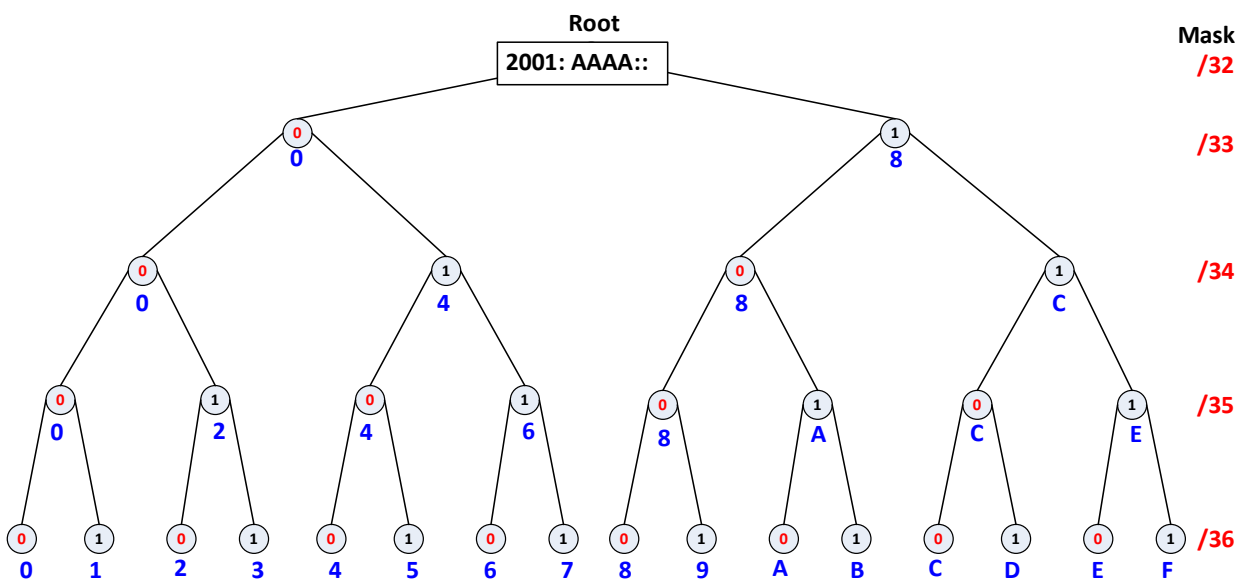


Figure 8.78

Right now, the tree is set up to start the subnetting.

From the observation of the tree, there are several options. For example, two blocks of mask /33 can be obtained like this:

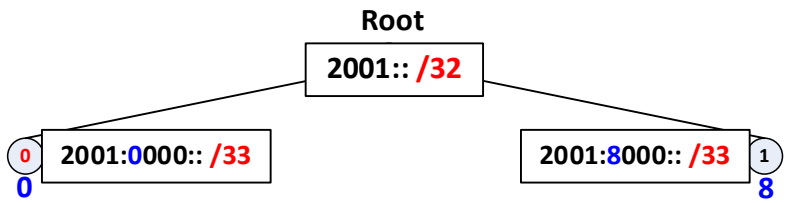
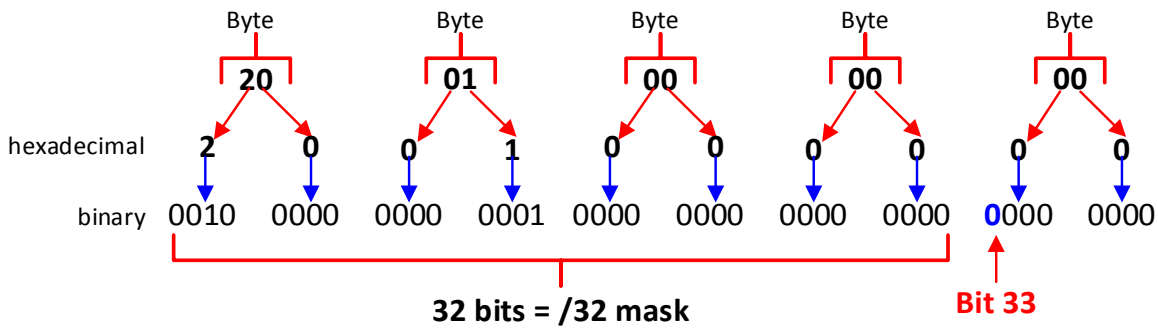
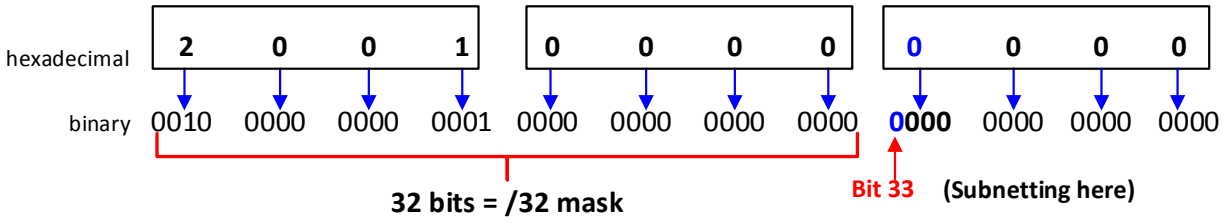


Figure 8.79

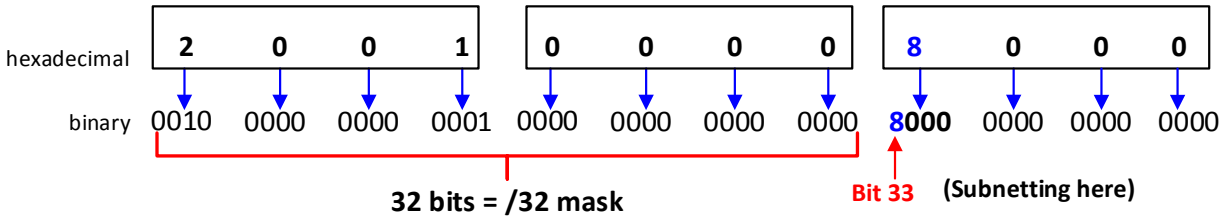
Alternatively, what just happened graphically, can also be done by working the binary combination:



The bit in position 33 can take up two values, either zero or one, and that yields either 0_{hex}



or the other possible value 8_{hex}



As it can be appreciated, even though the two methods yield the same result, it is obviously easier to use the graphical tree than the binary conversion with the added benefit that the tree allows for a quick visualization of the aggregations.

Continuing with the allocation, four blocks of mask /34 can be derived from the /33 blocks.

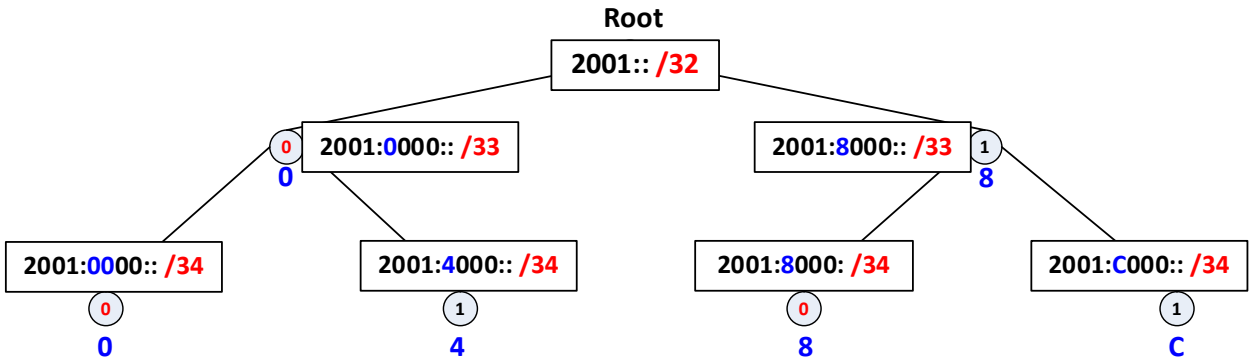


Figure 8.83

Where the prefix 2001:0000::/33 aggregates the two blocks underneath and the prefix 2001:8000:: /33 aggregates the other two blocks underneath.

The equivalent binary explanation is below:

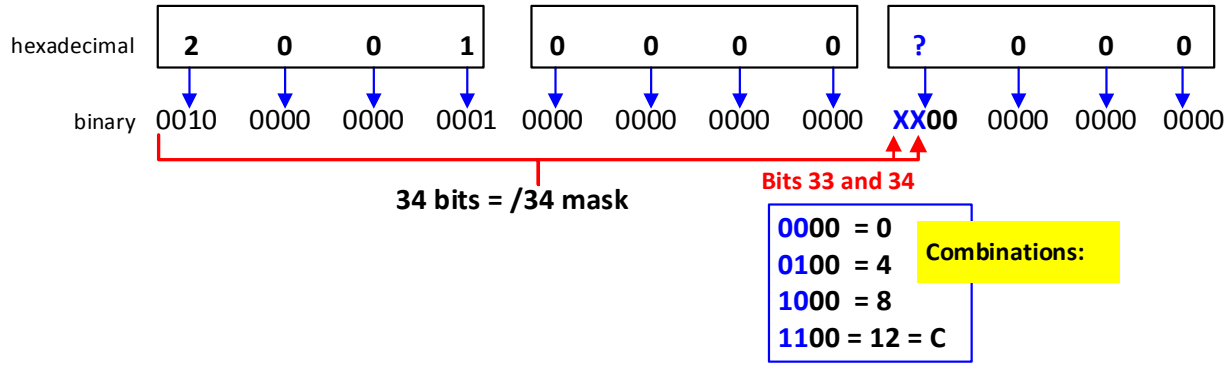


Figure 8.86

Furthermore, eight blocks of mask /35 are created next:

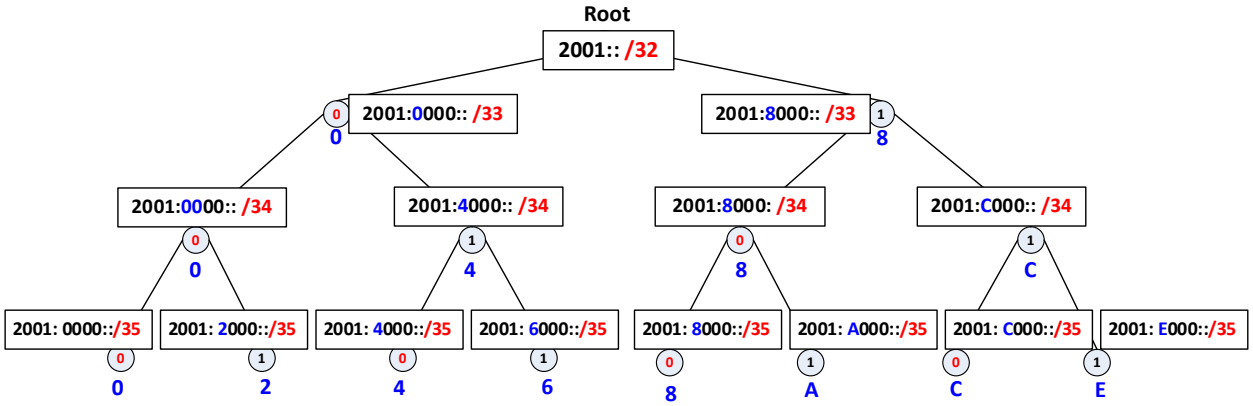


Figure 8.85

Finally, there are 16 blocks of mask /36 each to complete the subnetting of the 4 bits space:

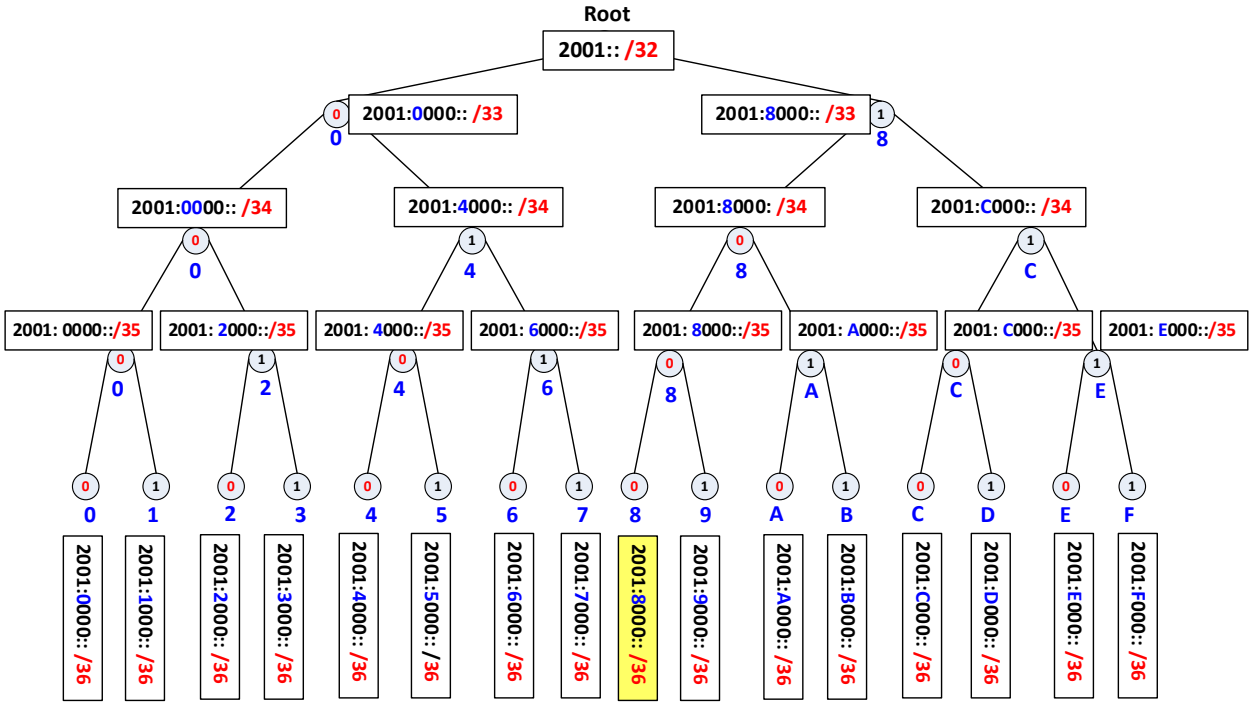
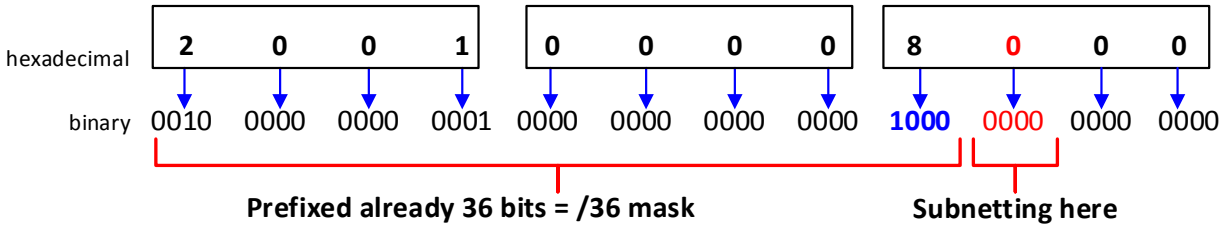


Figure 8.86

Example: this is the general idea. A large Internet carrier could receive the prefix 2001:: /32 (this range is actually advertised[1] by Hurricane Electric HE from AS 6939) then it would proceed to slice it into smaller pieces. For instance, take the block 2001:8000::/36 from the tree above and further subnet the next 4 bits available:



In this case, the preparation of the subnetting tree begins with the root:

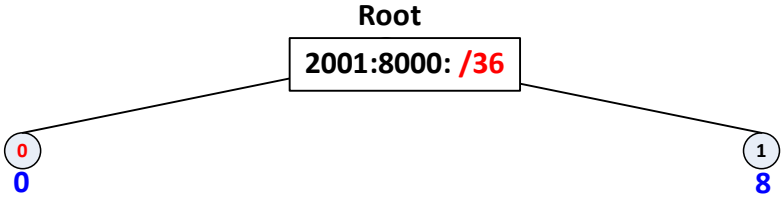


Figure 8.88

The first level of branches obtain these values:

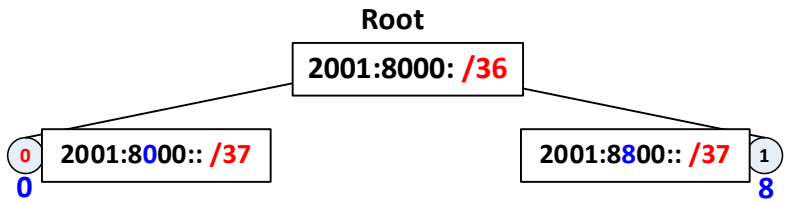


Figure 8.89

Then, by branching out each block, four blocks of mask /38 are made.

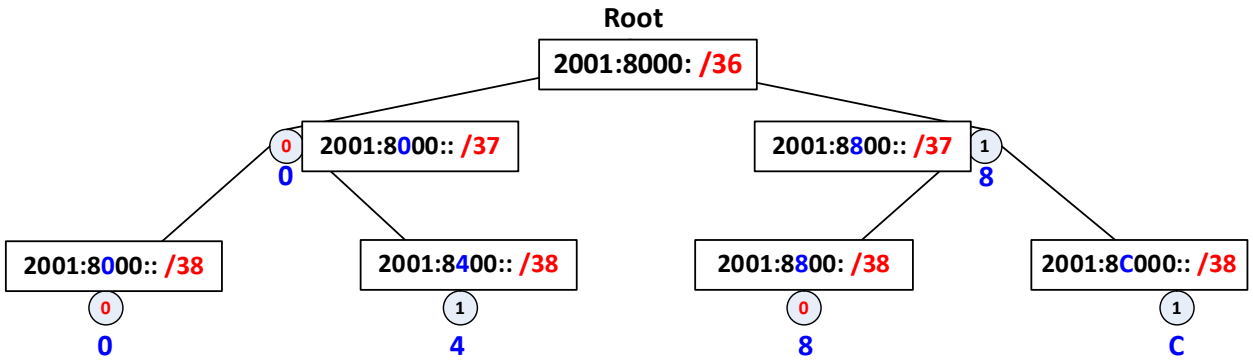


Figure 8.90

Continuing with another level of eight blocks of mask /39.

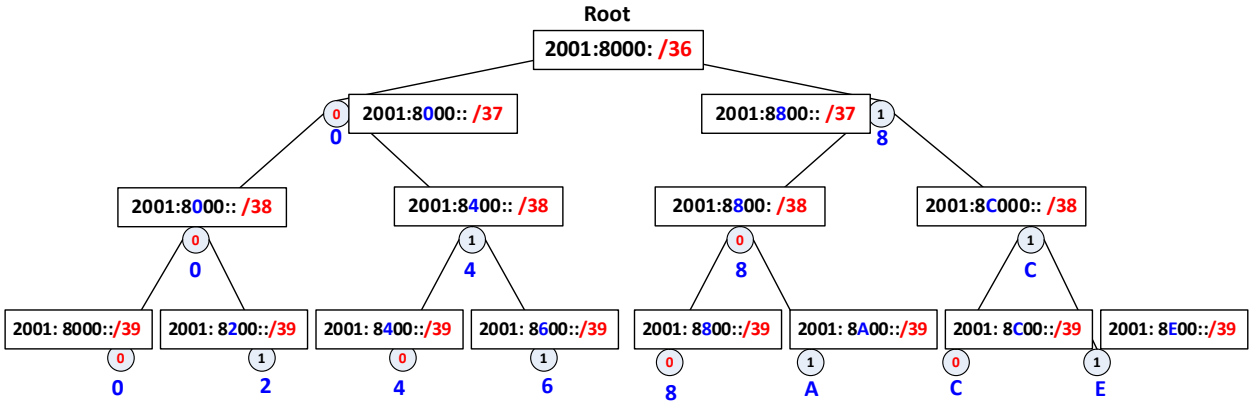


Figure 8.91

Finally, the 16 possible blocks created with the 4 bits of the hex coefficient:

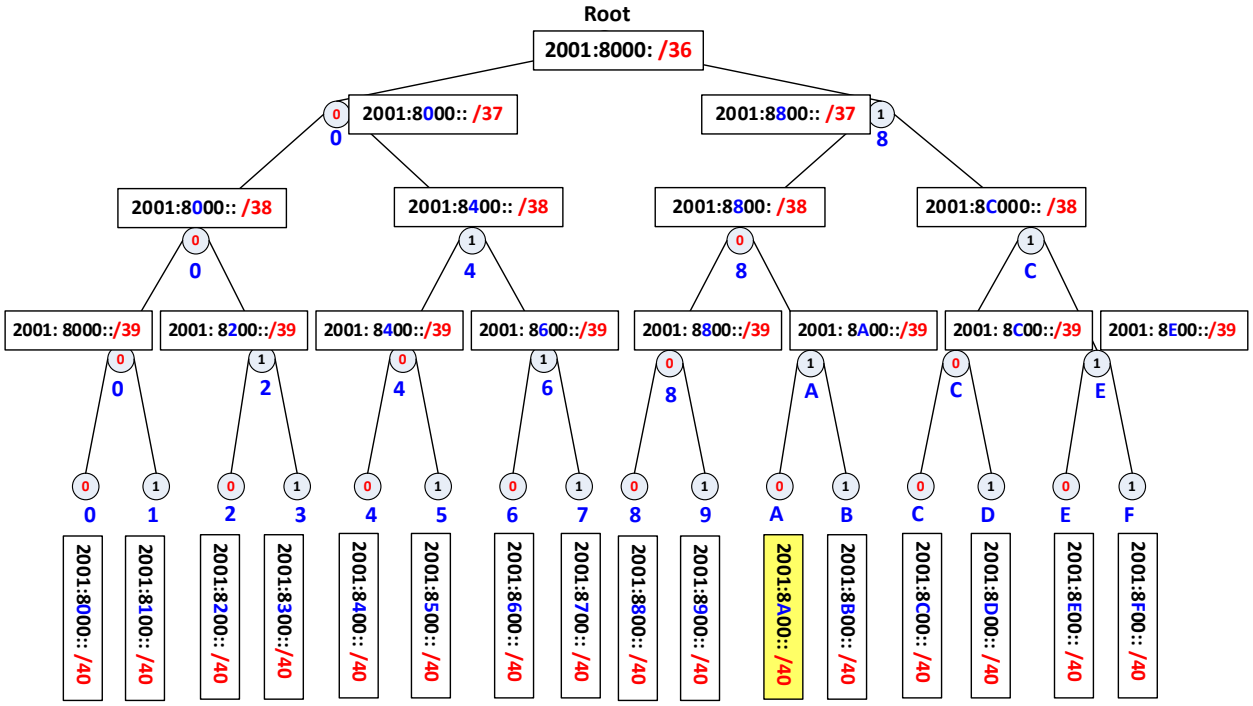


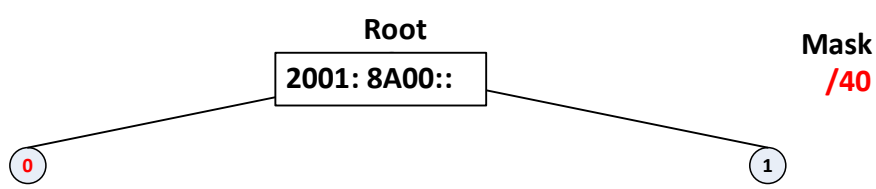
Figure 8.92

Even more, each one of these blocks can be sliced into smaller blocks, for example, take the block 2001:8A00::/40 and create 16 blocks of mask / 44.

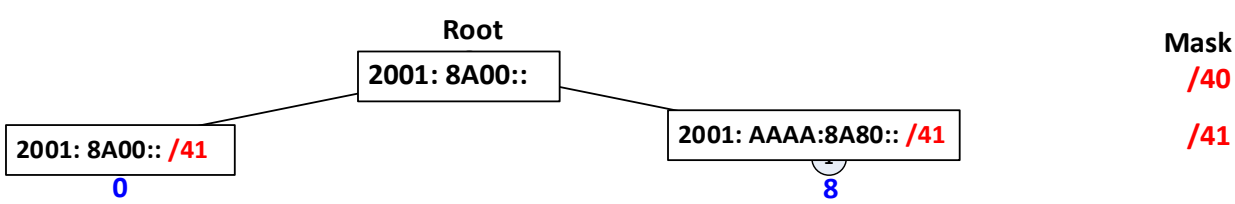
First) verify that the mask complies with the requirement $(mask = n \times 4)$

- $40 = 10 \times 4)$ so it will work

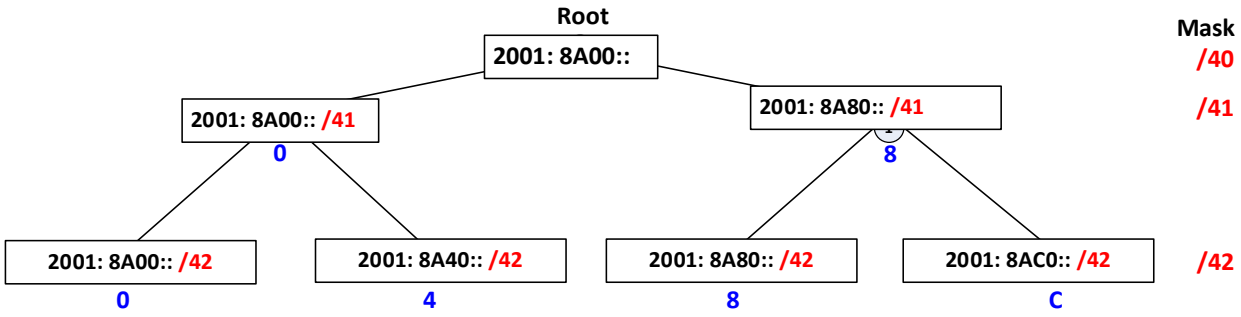
Second) set the prefix as the root of the tree:



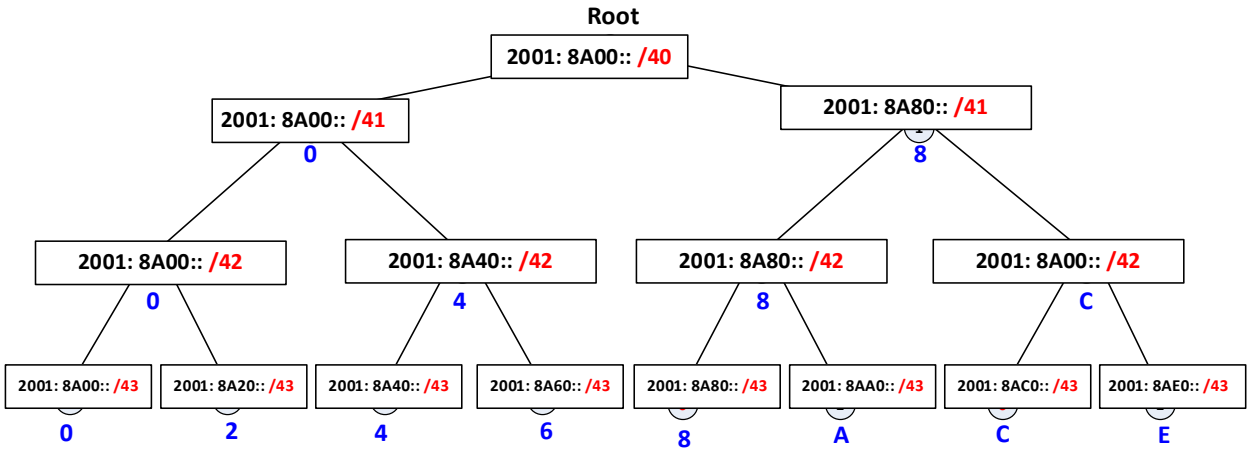
Third) start branching out to complete the 5 levels of the tree:



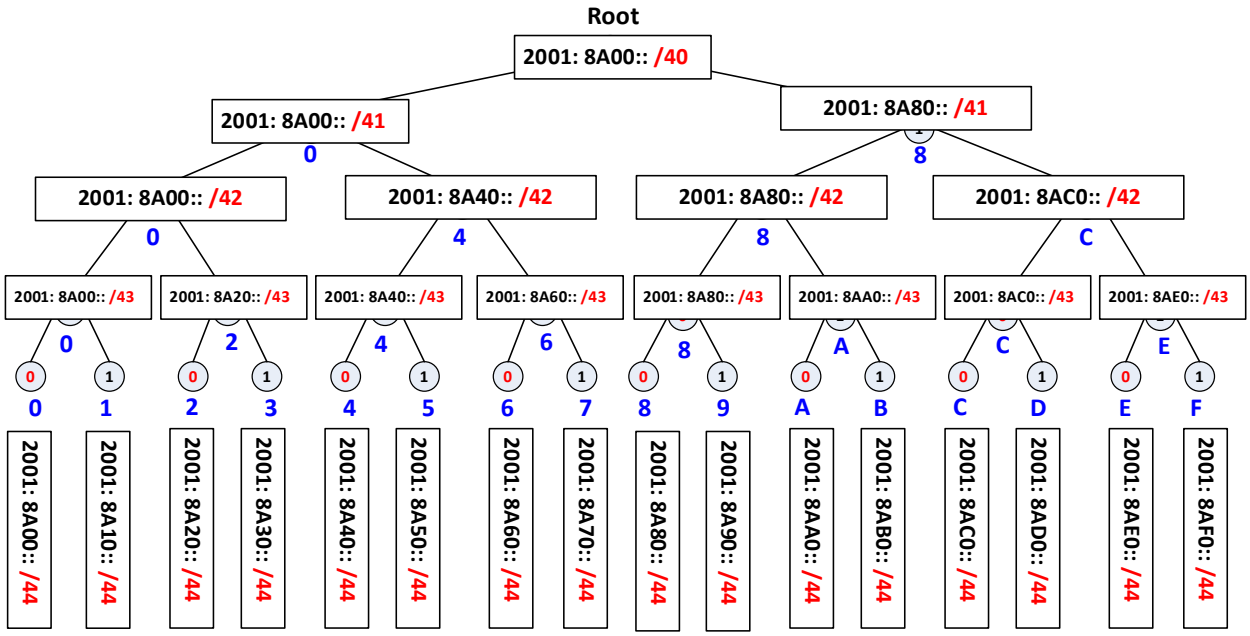
Fourth) find the next level for blocks size /42



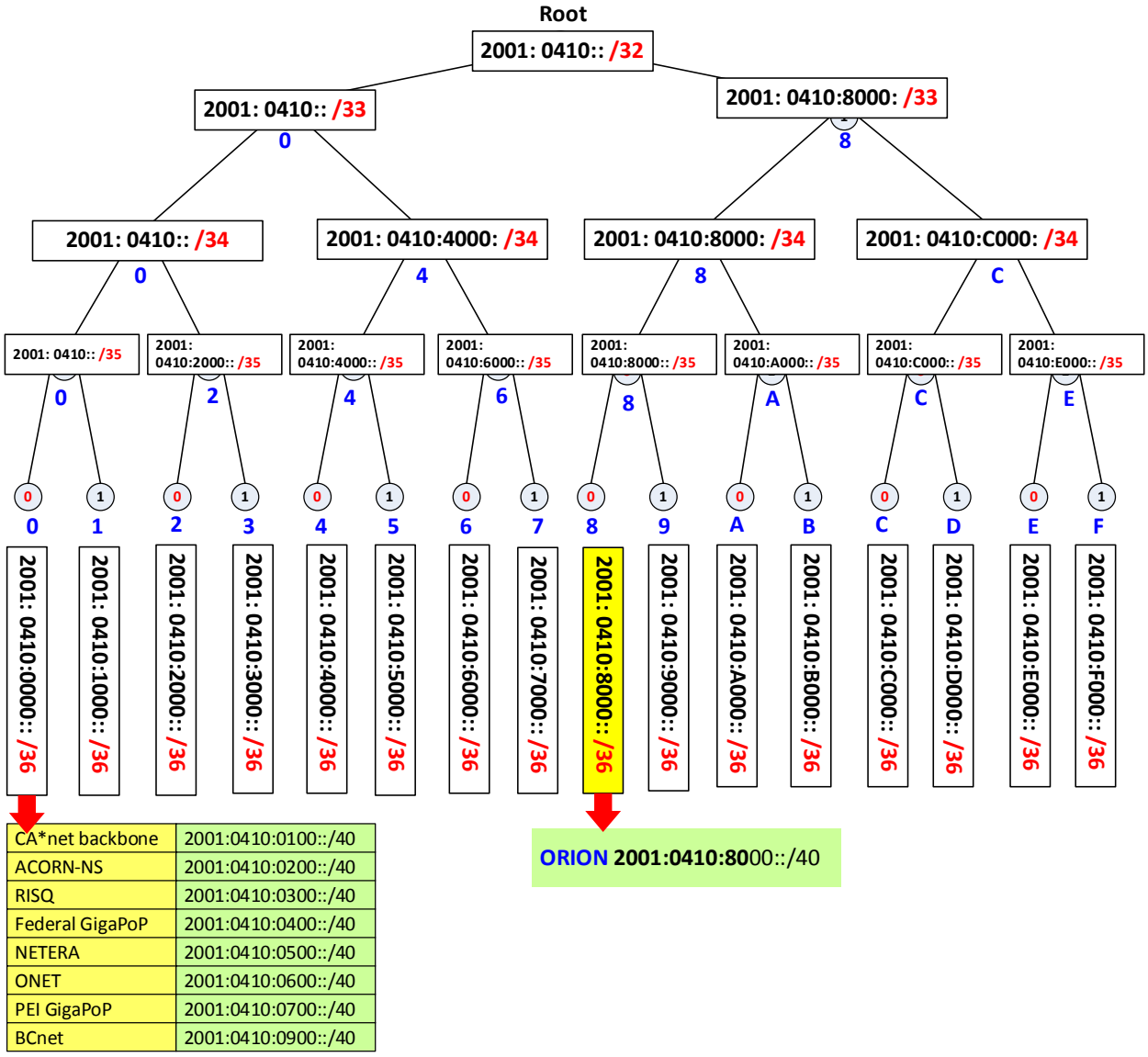
Fifth) Furthermore, eight blocks of mask //43:



Finally, 16 blocks of mask /36 each:



Example: AS 6509 Canarie [3] administers the IPv6 range 2001:0410::/32. From this space, Canarie carves out blocks to be allocated to its network partners.



CA*net backbone	2001:0410:0100::/40
ACORN-NS	2001:0410:0200::/40
RISQ	2001:0410:0300::/40
Federal GigaPoP	2001:0410:0400::/40
NETERA	2001:0410:0500::/40
ONET	2001:0410:0600::/40
PEI GigaPoP	2001:0410:0700::/40
BCnet	2001:0410:0900::/40

Figure 8.99

More subnetting can be performed on each /36 block until getting sixteen /40 blocks. For example, the table in the figure above shows [3] eight organizations receiving /40 address space from the block 2001:0410:0000::/36.

In the same way but from another block 2001:0410:8000::/36, Ontario's Orion (AS 26677) receives 2001:0410:8000::/40. Let's use this block as an example to see the details of how allocation of address space to final sites or customers can be done.

To reiterate, Canarie has sliced the block 2001:0410:8000::/36 into sixteen /40 blocks as shown below. One of the slices is given to Orion for its administration as summarized next:

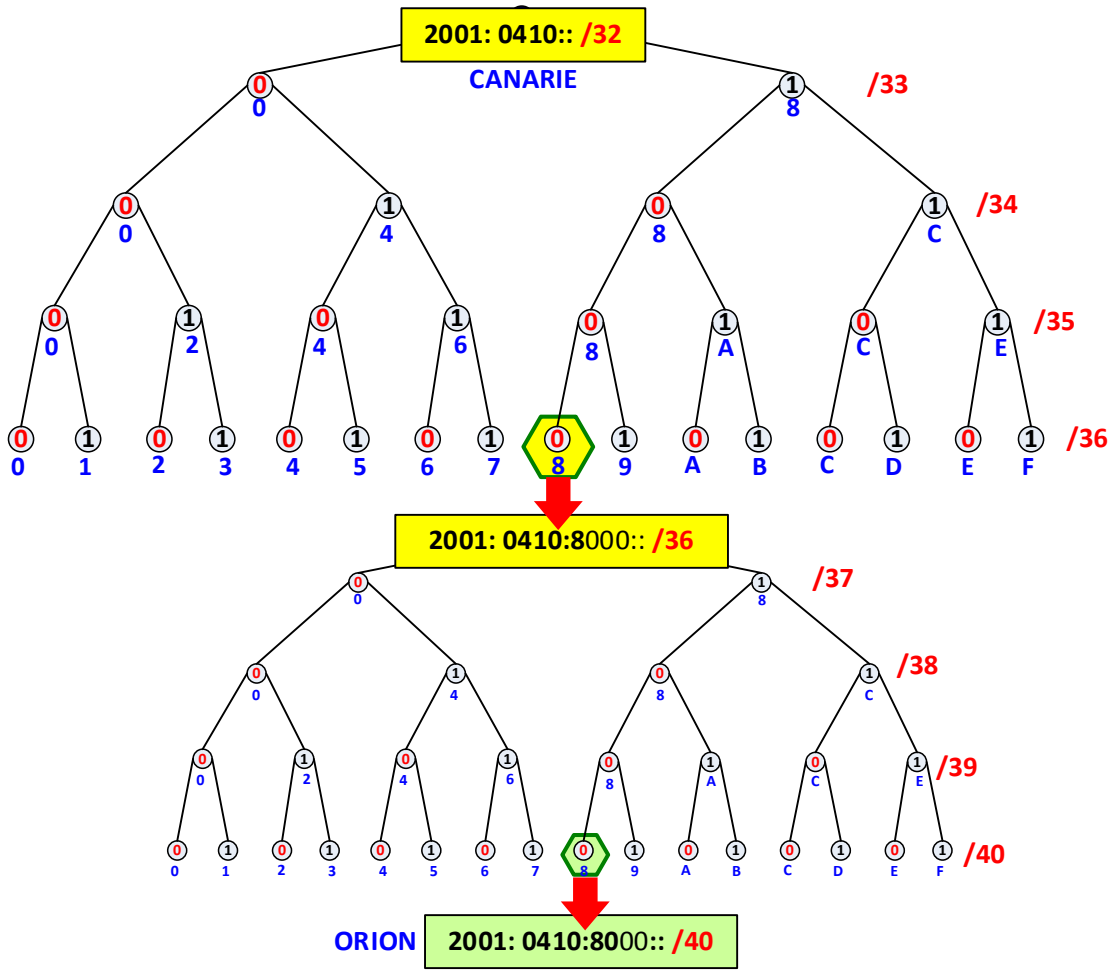


Figure 8.100

Orion keeps subnetting to make /48 blocks. From /40 to /48 there are $2^{8\ bits} = 256$ blocks, so Orion could have 256 partners (public institutions in Ontario) that receive /48 prefixes.

Each /48 is further sliced into /64 blocks. From /48 to /64 there are $2^{16\ bits} = 65{,}536$ blocks. These would be for the final local area networks. Essentially, a final customer like a University would receive a prefix /48 from Orion and make up to 65,536 LAN networks of /64 mask each. LAN networks with mask /64 are common practice to implement host address self-configuration where the MAC address of a computer or any device, gets embedded in the final 64 bits of the IPv6 address. This practice simplifies host address allocation although it is also a terrible waste of address space.

Configuration of BGP to Aggregate IPv6 address space

The following network diagram is meant to illustrate several levels of aggregation of IPv6 address space and the corresponding configuration of Cisco and Juniper routers.

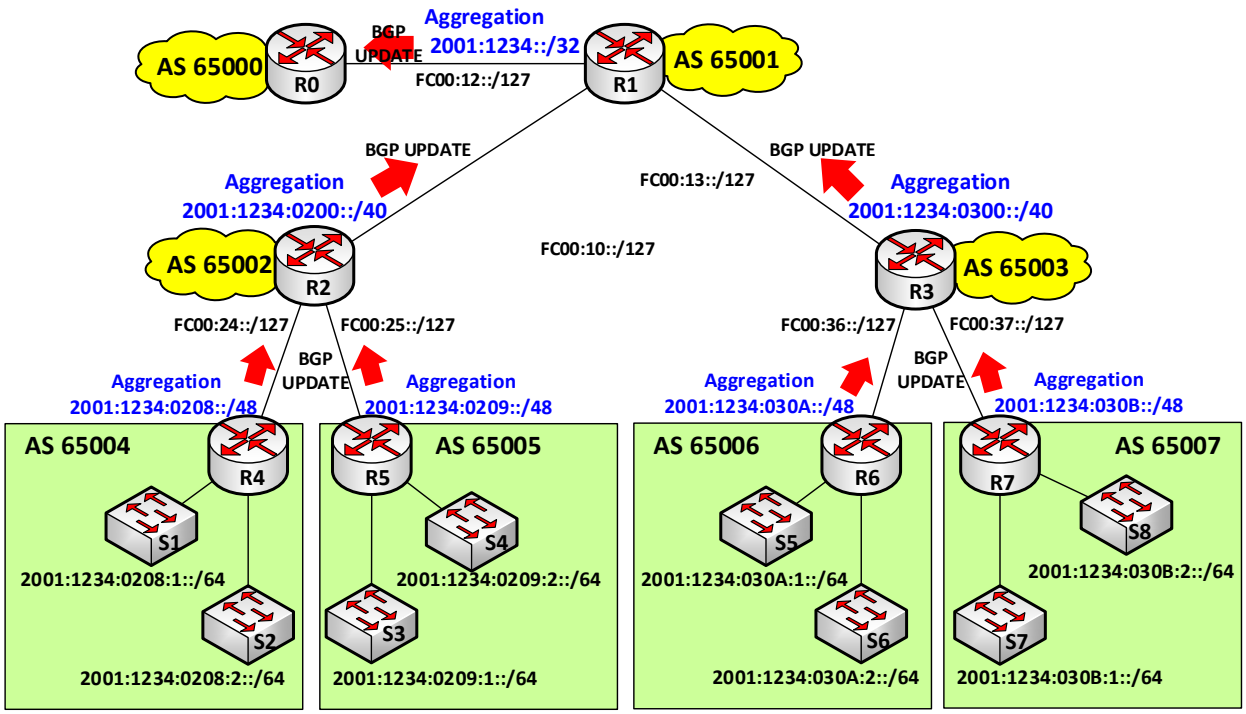
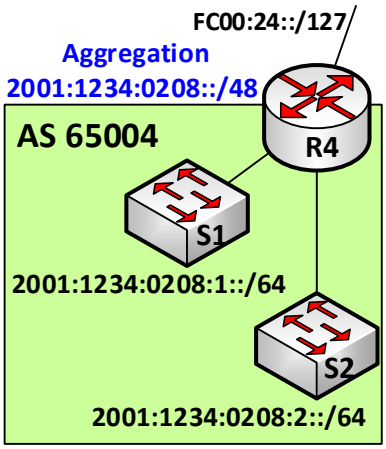


Figure 8.101

- Assume that AS 65001 is a carrier and it has been assigned the IPv6 address space 2001:1234::/32 from a regional Internet authority.
- Consequently, this AS is entitled to advertise the aggregation of the whole space to another carrier represented by AS 65000. Hence, 2001:1234::/32 will be the only information via BGP in AS65000's routing table.
- AS 65001 proceeds to splice its /32 space into multiple /40 blocks of addresses. There are 256 blocks /40 to be precise. Only two of those blocks are used here, but in principle, there could be 256 client ISPs receiving address space.
- AS 65002 is an Internet Service Provider (ISP2) that receives a block of addresses from AS65001. Such block's prefix is 2001:1234:0200::/40.
- AS 65003 is another Internet Service Provider (ISP32) that receives a block of addresses from AS65001. Such block's prefix is 2001:1234:0300::/40.

- AS65002 has, in turn, two final customers AS65004 and AS65005. Each receives a /48 block of addresses.
- AS65002 has also two final customers AS65006 and AS65007. Each receives a /48 block of addresses.
- The customers slice their assigned /48 space into /64 subnets. From /48 to /64 there are 16 bits available, so that is enough to create (2^{16}) 65,536 LAN networks each with /64 mask.
- Obviously, there is no point in advertising 65,536 IPv6 networks /64 to clutter the routing tables of the upstream routers, so each customer space is aggregated into the /48 blocks listed here.
- AS65004 advertises 2001:1234:0208::/48.
- AS65005 advertises 2001:1234:0209::/48.
- AS65006 advertises 2001:1234:030A::/48.
- AS65007 advertises 2001:1234:030B::/48.

Let's start the configuration with customer router R4 @AS65004. For practical reasons only the BGP configuration and only two subnets out of the possible 65,538 LANs are shown here.




```
router bgp 65004
 neighbor FC00:24:: remote-as 65002
 address-family ipv6
  network 2001:1234:208:1::/64
  network 2001:1234:208:2::/64
  aggregate-address 2001:1234:0208::/48 summary-only
  neighbor FC00:24:: activate
 exit-address-family
```

Figure 8.102

The suffix "summary-only" guarantees that BGP will only advertise the aggregation 2001:1234:0208::/48 without the more specific details.

Customer router R5 @AS65005 is quite similar to R4's configuration:

```
router bgp 65005
neighbor FC00:25:: remote-as 65002
address-family ipv6
 network 2001:1234:0209:1::/64
 network 2001:1234:0209:2::/64
 aggregate-address 2001:1234:0209::/48 summary-only
 neighbor FC00:25:: activate
exit-address-family
```

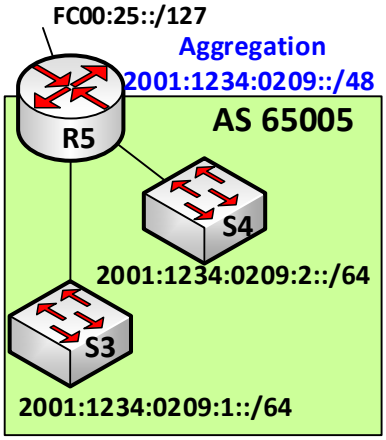
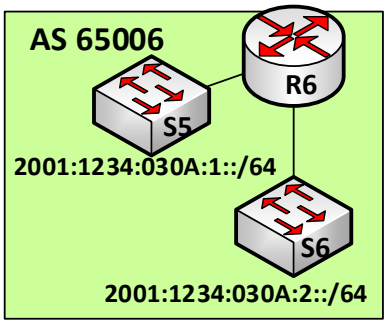


Figure 8.103

Router R6 is a Juniper router that aggregates 2001:1234:030A:0001::/64 and 2001:1234:030A:0002::/64 into 2001:1234:030A::/48.



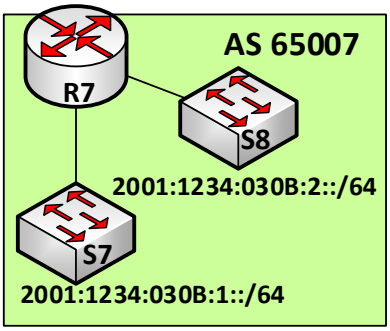
Aggregation
2001:1234:030A::/48

```
set interfaces em0 unit 0 family inet6 address 2001:1234:030A:1::1/64
set interfaces em1 unit 0 family inet6 address 2001:1234:030A:2::1/64
set interfaces em3 unit 0 family inet6 address fc00:36::1/127
set interfaces lo0 unit 0 family inet address 10.6.6.6
set routing-options rib inet6.0 aggregate route 2001:1234:30A::/48
set routing-options router-id 10.6.6.6
set routing-options autonomous-system 6
set protocols bgp group to-ISP-AS3 type external
set protocols bgp group to-ISP-AS3 export AGGREGATE/48
set protocols bgp group to-ISP-AS3 neighbor fc00:36::0 peer-as 3
set policy-options policy-statement AGGREGATE/48 term 1 from protocol aggregate
set policy-options policy-statement AGGREGATE/48 term 1 from route-filter 2001:1234:30A::/48 exact
set policy-options policy-statement AGGREGATE/48 term 1 then accept
set policy-options policy-statement AGGREGATE/48 term 2 then reject
```

Figure 8.104

Router R7 is also a Juniper router with configuration similar to R6:

Aggregation
2001:1234:030B::/48




```
set interfaces lo0 unit 0 family inet address 10.7.7.7/32
set interfaces em0 unit 0 family inet6 address 2001:1234:030B:1::1/64
set interfaces em1 unit 0 family inet6 address 2001:1234:030B:2::1/64
set interfaces em4 unit 0 family inet6 address fc00:37::1/127
set routing-options rib inet6.0 aggregate route 2001:1234:030b::/48
set routing-options router-id 10.7.7.7
set routing-options autonomous-system 7
set protocols bgp group to-ISP-AS3 type external
set protocols bgp group to-ISP-AS3 export AGGREGATE/48
set protocols bgp group to-ISP-AS3 neighbor fc00:37::0 peer-as 3
set policy-options policy-statement AGGREGATE/48 term 1 from protocol aggregate
set policy-options policy-statement AGGREGATE/48 term 1 from route-filter 2001:1234:030B::/48 exact
set policy-options policy-statement AGGREGATE/48 term 1 then accept
set policy-options policy-statement AGGREGATE/48 term 2 then reject
```

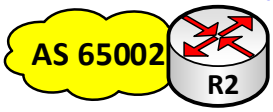
Figure 8.105

In Juniper, the aggregation must be manually defined to be inserted into the Routing Information Base (rib) for IPv6. One that the manual prefix exists in the routing table, the policy named AGGREGATE/48 matches the aggregation prefix and mask exactly. The policy does not accept anything else so both routers R6 and R7 only advertise the supernet for their respective LAN networks.

Notice that the aggregations /48 cover all the /64s for each site, not just the ones shown in the diagram. Obviously, it would not be feasible to show them all in this explanation.

```
router bgp 65002
 neighbor FC00:12:: remote-as 65001
 neighbor FC00:24::1 remote-as 65004
 neighbor FC00:25::1 remote-as 65005
!
 address-family ipv6 unicast
  default-information originate
  neighbor FC00:12:: activate
  aggregate-address 2001:1234:200::/40 as-set
  neighbor FC00:12:: route-map AGGREGATE/40 out
  neighbor FC00:24::1 activate
  neighbor FC00:24::1 default-originate
  neighbor FC00:24::1 route-map DEFAULT out
  neighbor FC00:25::1 activate
  neighbor FC00:25::1 default-originate
  neighbor FC00:25::1 route-map DEFAULT out
 exit-address-family
!
route-map DEFAULT permit 10
 match ipv6 address prefix-list DEFAULT
!
ipv6 prefix-list DEFAULT seq 5 permit ::/0
!
route-map AGGREGATE/40 permit 10
 match ipv6 address prefix-list AGG/40
!
ipv6 prefix-list AGG/40 seq 5 permit 2001:1234:200::/40
!
```

Aggregation
2001:1234:0200::/40

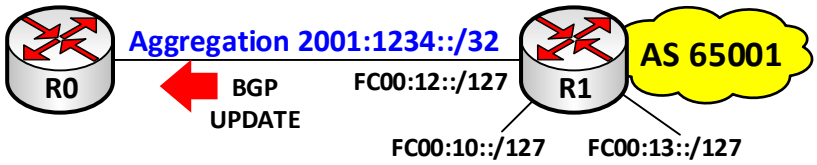


Router R2 advertises to R1 one aggregation /40 to cover all the space of its customers thus reducing the size of the routing table of the upstream carrier.

Router R2 advertises an IPv6 default route **::/0** to both clients. This is optional, but in the current topology, it is the logical solution. The clients only receive one BGP entry from the provider instead of a full routing table.

Figure 8.106

Router R1 @65001 is a Juniper router. It performs /32 level aggregation.




```
set interfaces em0 unit 0 family inet6 address fc00:0013::0/127
set interfaces em1 unit 0 family inet6 address fc00:0010::1/127
set interfaces em3 unit 0 family inet6 address fc00:0012::0/127
set interfaces lo0 unit 0 family inet address 10.1.1.1/32

set routing-options rib inet6.0 aggregate route 2001:1234::/32
set routing-options router-id 10.1.1.1
set routing-options autonomous-system 65001

set protocols bgp group to-AS65002 type external
set protocols bgp group to-AS65002 neighbor fc00:0012::1 peer-as 65002

set protocols bgp group to-AS65003 type external
set protocols bgp group to-AS65003 neighbor fc00:0013::1 peer-as 65003

set protocols bgp group to-AS65000 type external
set protocols bgp group to-AS65000 export ADV-AGGREGATION/32
set protocols bgp group to-AS65000 export DENY-ELSE
set protocols bgp group to-AS65000 neighbor fc00:0010::0 peer-as 65000

edit policy-options policy-statement ADV-AGGREGATION/32
  set term 1 from protocol aggregate route-filter 2001:1234::/32 exact
  set term 1 then accept

set policy-options policy-statement DENY-ELSE term 1 then reject
```

An aggregate entry for the /32 prefix is created under a routing option.

This route is accepted by the routing policy.

BGP is configured to export the result of the policy to R0 in AS65000.

Consequently, R0 receives from R1 the /32 prefix only.

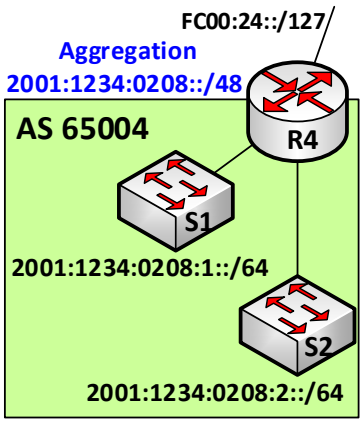
Figure 8.107

Juniper router J3 advertises an aggregation /40 to cover all of the /48 from the customer sites AS65006 and AS65007 (and many more not shown). It also generates a default route ::/0 to its customer sites.

```
root@J3> show configuration | display set

  set system host-name J3
  set interfaces em0 unit 0 family inet6 address fc00:13::1/127
  set interfaces em3 unit 0 family inet6 address fc00:36::/127
  set interfaces em4 unit 0 family inet6 address fc00:37::/127
  set interfaces lo0 unit 0 family inet address 10.3.3.3/32

set routing-options rib inet6.0 aggregate route 2001:1234:0300::/40
set routing-options rib inet6.0 generate route ::/0 discard
set routing-options router-id 10.3.3.3
set routing-options autonomous-system 65003
set protocols bgp group to-AS65001 type external
set protocols bgp group to-AS65001 neighbor fc00:13::0 peer-as 65001
set protocols bgp group to-AS65001 export [ AGGR/40 DENY-ELSE ]

set protocols bgp group to-AS65006 type external
set protocols bgp group to-AS65006 neighbor fc00:36::1 peer-as 65006
set protocols bgp group to-AS65006 export [ IPv6-DEFAULT DENY-ELSE ]

set protocols bgp group to-AS65007 type external
set protocols bgp group to-AS65007 neighbor fc00:37::1 peer-as 65007
set protocols bgp group to-AS65007 export [ IPv6-DEFAULT DENY-ELSE ]

set policy-options policy-statement AGGR/40 term 1 from protocol aggregate
set policy-options policy-statement AGGR/40 term 1 from route-filter 2001:1234:300::/40 exact
set policy-options policy-statement AGGR/40 term 1 then accept

set policy-options policy-statement IPv6-DEFAULT term 1 from route-filter ::/0 exact
set policy-options policy-statement IPv6-DEFAULT term 1 then accept

set policy-options policy-statement DENY-ELSE then reject
```

Figure 8.108

Once that the configurations of all the routers are completed the results are going to be analyzed from the bottom of the network going up starting with routers R4 and R5.




```
R4#show bgp ipv6 unicast
Local router ID is 10.4.4.4
   Network                  Next Hop     Path
*  ::/0                     FC00:24::    65002 i
*> 2001:1234:208::/48       ::                 i
*  2001:1234:208:1::/64     ::                 i
*  2001:1234:208:2::/64     ::                 i
```

Router R4 receives a default IPv6 route ::/0 from its Internet provider. In this topology, that is enough to have connectivity.

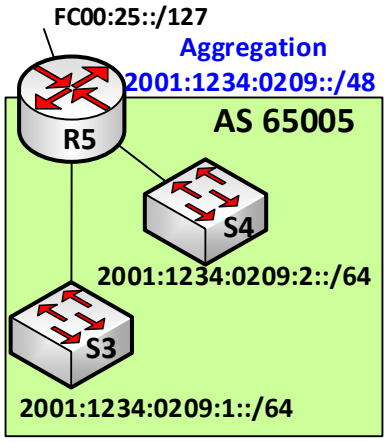
R4 creates an aggregation /48 that covers all of the /64 subnets inside AS 65004.

Two of the /64 are shown in the table but in principle there could be 65,536 /64 subnets in this site.

R4 only advertises the /48 aggregation to the upstream router R2 in AS 65002.

Figure 8.109

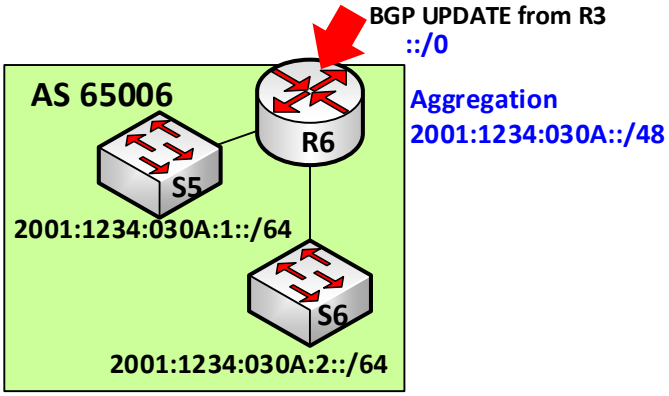
Router R5 has a similar situation than R4 but for the block 2001:1234:0209::/48.




```
R5#show bgp ipv6 unicast
Local router ID is 10.5.5.5
   Network                  Next Hop     Path
*  ::/0                     FC00:24::    65002 i
*> 2001:1234:209::/48       ::                 i
*  2001:1234:209:1::/64     ::                 i
*  2001:1234:209:2::/64     ::                 i
```

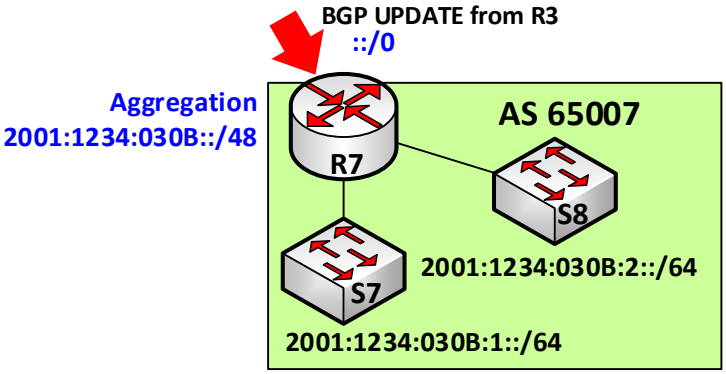
Figure 8.110

Both routers R6 and R7 advertise /48 aggregations while they receive a default ::/0 from R3.



R6#show route terse

Network	Protocol	Preference	Metric 1	Next Hop	Path
* ::/0	B	170	100	>FC00:36::	65003 i
* 2001:1234:30A::/48	A	130		Reject	
* 2001:1234:30A:1::/64	D	0		>em0.0	
* 2001:1234:30A:2::/64	D	0		>em1.0	

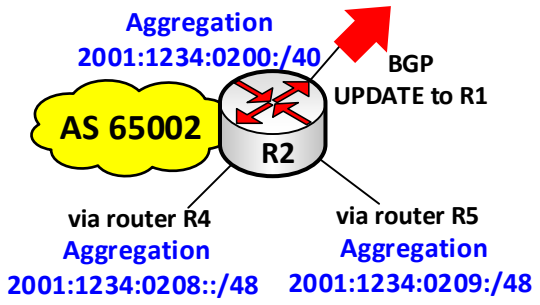


R7#show route terse

Network	Protocol	Preference	Metric 1	Next Hop	Path
* ::/0	B	170	100	>FC00:37::	65003 i
* 2001:1234:30B::/48	A	130		Reject	
* 2001:1234:30B:1::/64	D	0		>em0.0	
* 2001:1234:30B:2::/64	D	0		>em1.0	

Figure 8.112

Router R2 shows in its routing table an aggregation 2001:1234:000::/40 that covers the space of for the two customers AS 65004 and AS 65005. The entry has an AS-SET {65004, 65005} to show that it is an aggregation for two different autonomous systems. R2 advertises such aggregation to the upstream router R1 in AS65001.



R2#show bgp ipv6 unicast (edited)

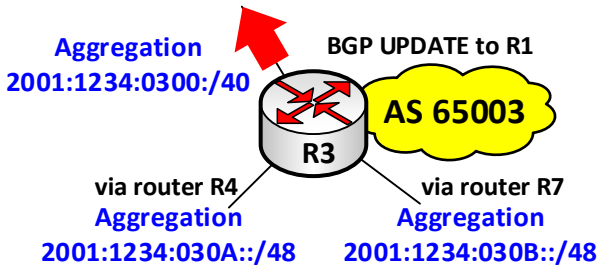
Network	Next Hop	Path
*> 2001:1234:0200::/40	::	{65004, 65005} i
*> 2001:1234:0208::/48	FC00:24::1	65004 i
*> 2001:1234:0209::/48	FC00:25::1	65005 i
*> 2001:1234:0300::/40	FC00:12::	65001 65003 {65006, 65007} i

Figure 8.113

Also, router R2 knows the more specific prefix 2001:1234:0208::/48 advertised directly by router R4 in AS 65004 and prefix 2001:1234:0209::/48 advertised directly by router R5 in AS 65005. Notice that the details of the customer sites are hidden by the aggregations which is the main point of this arrangement.

Finally, router R2 have also learned the aggregation 2001:1234:0300::/40 from R1. Such aggregation covers the address space of AS 65006 and AS 65007 as the AS-SET {65006, 65007} clearly indicates.

Router R3's routing table contains the aggregation 2001:1234:0200::/40 made by router R2 in AS65002 to cover the address space of its customers AS65004 and AS65005.



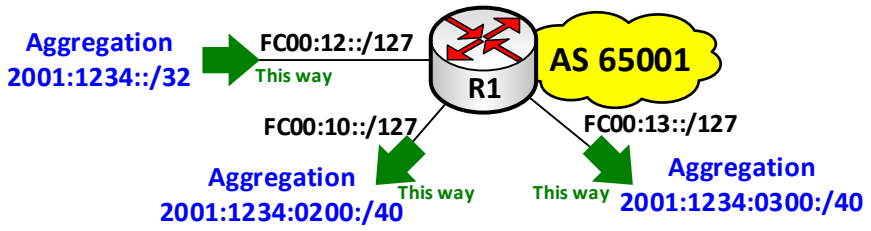
R3#show route terse (edited)

Network	Next Hop	AS Path
* 2001:1234:0200::/40	FC00:13::	65001 65002 {65004, 65005} i
* 2001:1234:300::/40	Reject	
* 2001:1234:030A::/48	FC00:36::1	65006 i
* 2001:1234:030B::/48	FC00:37::1	65007 i

Figure 8.114

Router R3 knows the two /48 aggregations advertised by its customer sites AS65006 and AS65007. Therefore, R3 creates its own aggregation 2001:1234:0300::/40 to cover such address space which is then advertised to R1.

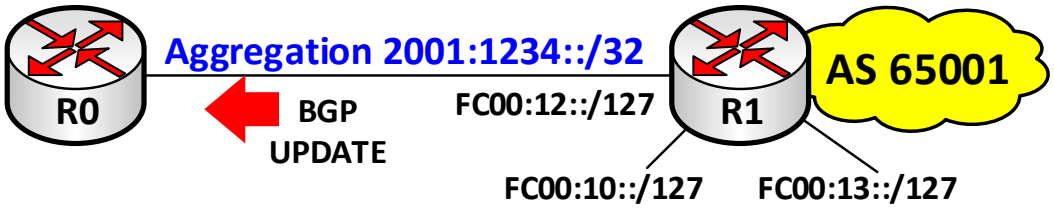
Router R1 only has three prefixes in its routing table (beside the directly connected networks). The hierarchical aggregation eliminates unnecessary clutter in this router and routing decisions can be made more efficiently.




```
root@J1> show route terse

A Destination            P Prf  Metric 1  Metric 2  Next hop     AS path

* 2001:1234::/32          A 130                      Reject

* 2001:1234:200::/40      B 170     100       0      >fc00:12::1    65002 { 65004  65005} i

* 2001:1234:300::/40      B 170     100              >fc00:13::1    65003  { 65006  65007} i
```

Figure 8.115

Finally, this concept is taken to the highest level when R0 only receives the /32 aggregation (which contains 79,228,162,514,264,337,593,543,950,336 addresses). Interestingly, notice the AS-SET that contains all the customer AS numbers.




```
R0#show bgp ipv6 unicast (edited)

Network                Next Hop    Path

*> 2001:1234::/32      FC00:10::1  65001 {65002, 65003, 65004, 65005, 65006, 65007} i
```

Figure 8.116

A sniff of a BGP UPDATE helps to analyze in more detail. Router R1@AS65001 sends the aggregation 2001:1234:: /32 only to R0 @AS65000. Notice that the AS path contains a sequence initiated by AS 65001 (where the aggregation /32 happens) that continues with the AS-SET of all the AS numbers that participate in the address space being aggregated.

Border Gateway Protocol (edited for simplicity)

Marker FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

Length 119 bytes

TYPE: UPDATE message (2) (from R1 to R0)

Unfeasible routes length 0 bytes

Total path attributes length 96 bytes

Path attributes:

ORIGIN: IGP

AS_PATH: 65001 {65002, 65003, 65004, 65005, 65006, 65007}

AGGREGATOR: AS: 65001 origin 10.1.1.1 (this is R1@AS65001)

MP_REACH_NLRI:

2001:1234:: /32

Figure 8.117

Chapter Summary

In this chapter, the principles of the aggregation of IP address space have been explained from two viewpoints. First, a technique to allocate IP address space efficiently has been presented. This method is based upon a binary tree and it is meant to be a tool for the network engineer when administering address blocks. Second, aggregation was implemented by configuring routers showing both the Cisco and Juniper ways. Several cases were presented.

Aggregation has a very important role in IPv6 deployments. Consequently, this chapter also covered the administration principles of IPv6 address space and its practical configuration.

Exercises

Exercise 8.1

An ISP AS23 has the address block 24.0.128.0/17 available for its customers in the topology.

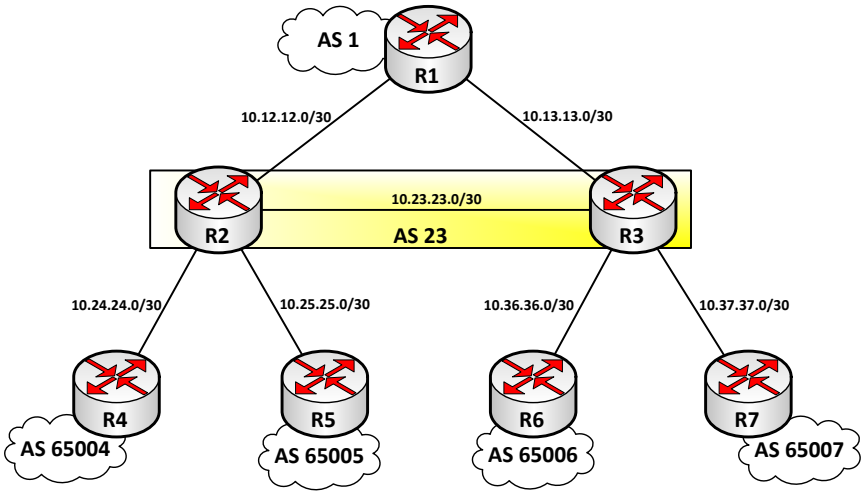


Figure 8.118

The address space requirements are the following:

Location	Requirement	IP prefix	Mask
AS 65004	Contains address space of 16 networks /24		
AS 65005	Contains address space of 16 networks /24		
AS 65006	Contains address space of 32 networks /24		
AS 65007	Contains address space of 64 networks /24		
R2	Advertises to R1 aggregation to cover AS65004 and AS65005		
R5	Advertises to R1 aggregation to cover AS65006 and AS65007		

Exercise 8.2

Allocate the address space 64.128. /9 according to the specific requirements in the table.

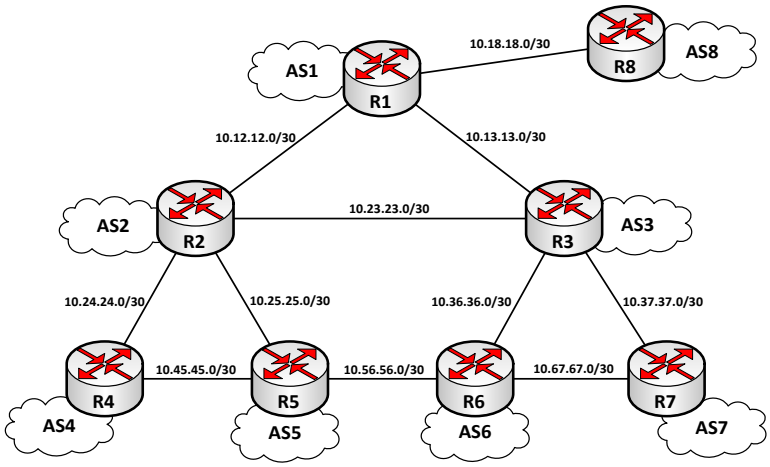


Figure 8.119

Location	Requirement	IP prefix	Mask
AS1	Contains the IP address space of 64.128.0.0/10	64.128.0.0	/10
AS2	Contains 16 blocks of IP addresses with mask /16 each		
AS3	Contains 16 blocks of IP addresses with mask /16 each		
AS4	Contains 8 blocks of IP addresses with mask /16 each		
AS5	Contains 8 blocks of IP addresses with mask /16 each		
AS6	Contains space of 524,288 IP addresses		
AS7	Contains space of 524,288 IP addresses		
R2	Advertises to R3 aggregation that covers AS5 and AS4		
R3	Advertises to R2 aggregation that covers AS6 and AS7		
R1	Advertises to R8 aggregation for AS4, As5, AS6, and AS7		
R1	Advertises to R8 aggregation for AS2 and AS3		

Exercise 8.3

The following figure shows an Internet Service Provider AS 144 which provides connectivity to two private customers AS65001 and AS65002. The clouds inside the autonomous systems represent routing domains, areas, or just networks. The ISP owns the address space 24.0.0.0 /8 to be allocated according to the requirements of each network-cloud. Additionally, there are aggregation requirements in different parts of the topology. These requirements are presented in the tables below.

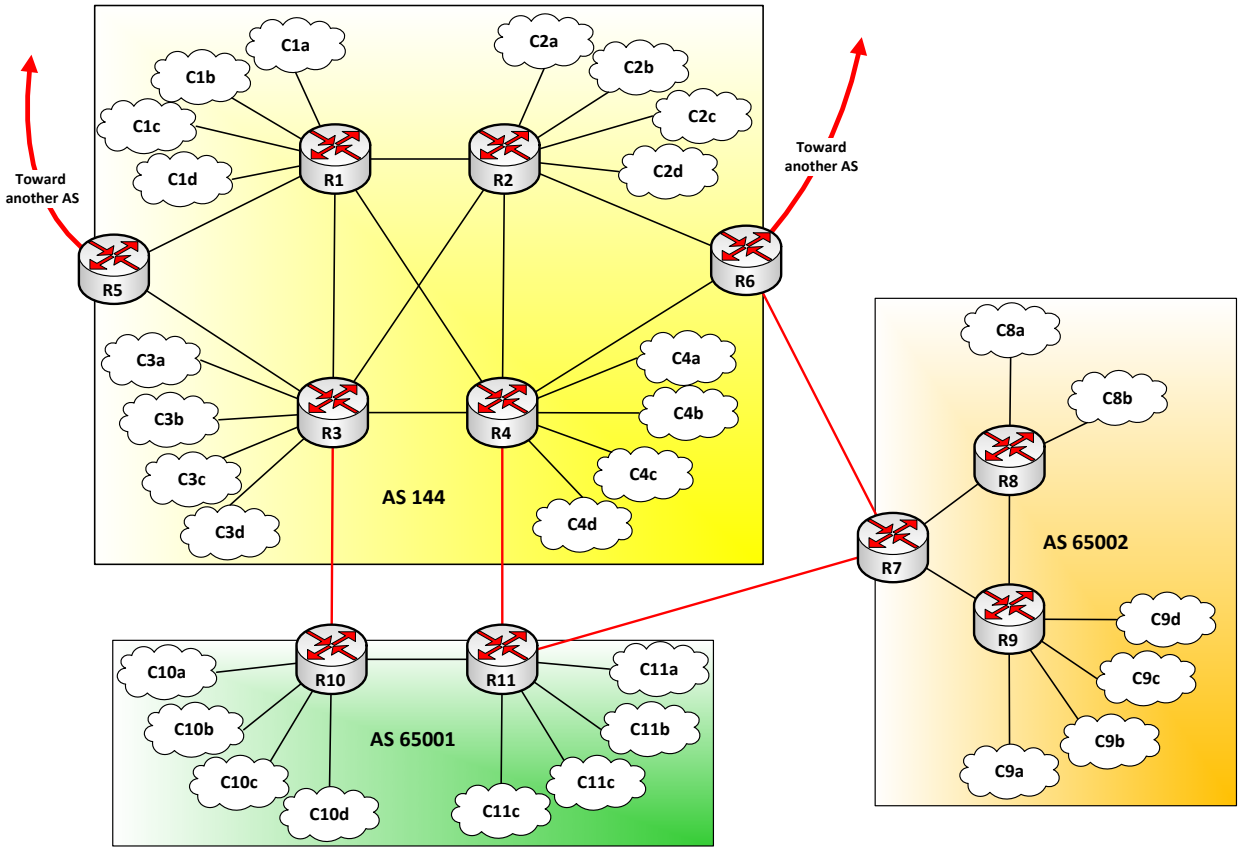


Figure 8.120

Autonomous System AS 144

Location	Requirement	IP prefix	Mask
C1a	Contains address space of 64 networks /24		
C1b	Contains address space of 64 networks /24		
C1c	Contains address space of 128 networks /24		
C1d	Contains address space of 256 networks /24		
R1	Advertises aggregation to cover all C1s clouds		
Location	Requirement	IP prefix	Mask
C2a	Contains address space of 32 networks /24		

C2b	Contains address space of 32 networks /24		
C2c	Contains address space of 64 networks /24		
C2d	Contains address space of 128 networks /24		
R2	Advertises aggregation to cover all C2s clouds		

Location	Requirement	IP prefix	Mask
C3a	Advertises space of 8,192 IP addresses		
C3b	Advertises space of 8,192 IP addresses		
C3c	Advertises space of 16,384 IP addresses		
C3d	Advertises space of 32,768 IP addresses		
R3	Advertises aggregation to cover all C3s clouds		

Location	Requirement	IP prefix	Mask
C4a	Advertises space of 256 networks /24		
C4b	Advertises space of 256 networks /24		
C4c	Advertises space of 256 networks /24		
C4d	Advertises space of 256 networks /24		
R4	Advertises aggregation to cover all C4s clouds		

Location	Requirement	IP prefix	Mask
R5	Advertises aggregation to cover all of AS 144		
R6	Advertises aggregation to cover all of AS 144		

Autonomous System AS 65001

Location	Requirement	IP prefix	Mask
C10a	Advertises space of 128 networks /24		
C10b	Advertises space of 128 networks /24		
C10c	Advertises space of 128 networks /24		
C10d	Advertises space of 32,768 IP addresses		
R10	Advertises aggregation to cover all C10s clouds		

Location	Requirement	IP prefix	Mask
C11a	Advertises space of 128 networks /24		
C11b	Advertises space of 128 networks /24		
C11c	Advertises space of 128 networks /24		
C11d	Advertises space of 32,768 IP addresses		
R11	Advertises aggregation to cover all C11s clouds		

Autonomous System AS 65002

Location	Requirement	IP prefix	Mask
C8a	Advertises space of one network /16		
C8b	Advertises space of 256 networks /24		
R8	Advertises aggregation to cover all C8s clouds		

Location	Requirement	IP prefix	Mask
C9a	Advertises space of 128 networks /24		
C9b	Advertises space of 128 networks /24		
C9c	Advertises space of 128 networks /24		
C9d	Advertises space of 128 networks /24		
R9	Advertises aggregation to cover all C9s clouds		

Location	Requirement	IP prefix	Mask
C2a	Contains address space of 32 networks /24		
C2b	Contains address space of 32 networks /24		
C2c	Contains address space of 64 networks /24		
C2d	Contains address space of 128 networks /24		
R2	Advertises aggregation to cover all C2s clouds		

Exercise 8.4

- Configure IPv6 and BGP in the following network topology.

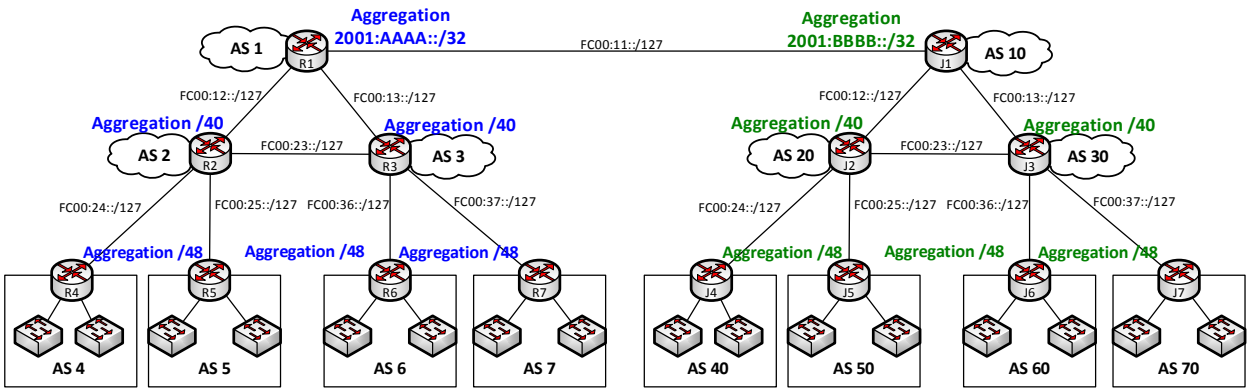


Figure 8.121

- The left is to be done with Cisco routers.
- Use address space 2001:AAAA::/32.
- Assign address space /64 to the LAN networks.
- The address allocation has to aggregate.

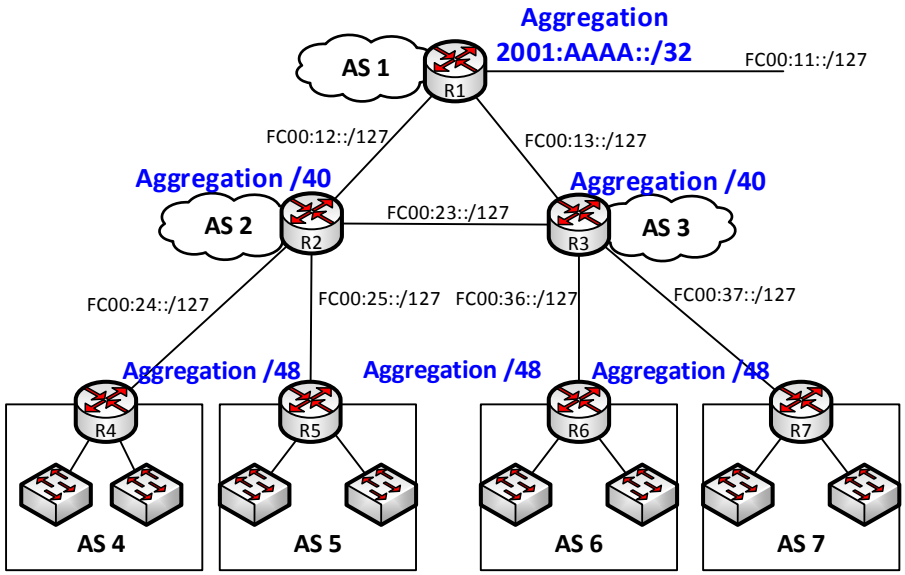


Figure 8.122

- The right is to be done with Juniper routers.
- Use address space 2001:BBBB::/32.
- Assign address space /64 to the LAN networks.
- The address allocation has to aggregate.

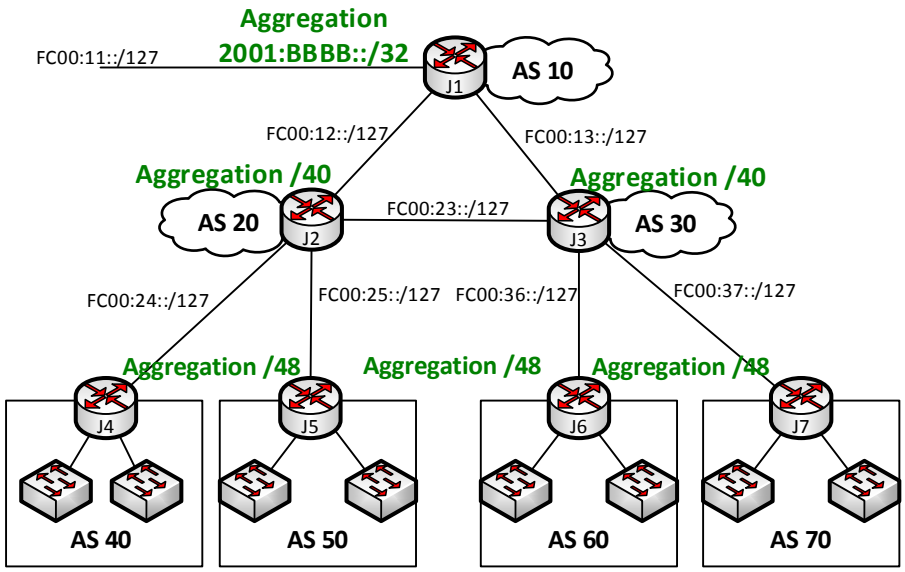


Figure 8.123

- Complete the network and test it.

Chapter References

[1] CIDR Report. http://www.cidr-report.org

[2] Internet Assigned Numbers Authority. https://www.iana.org/numbers. (July 2017)

[3] CANARIE. CA*net 4 IPv6 Address Allocation and Assignment Policy. (2003)

[4] R. Hinden. S. Deering. Internet Protocol Version 6 (IPv6) Addressing Architecture. RFC-3513. April 2003.

[5] R. Hinden, S. Deering, E. Nordmark. IPv6 Global Unicast Address Format. RCF - 3587. August 2003.

CHAPTER 9

Enterprise Access to Internet

This chapter deals with how an organization can connect to the Internet using BGP. Fundamentally, an enterprise network receives access from an Internet Service Provider ISP and both sides apply policies to control the traffic patterns. There are variations to the way that organizations use to access the Internet. The different techniques are individually explored in the chapter and afterwards they are integrated in a sample network.

Client Enterprise gets simple Internet Access from ISP

The first case to be demonstrated is an enterprise network with a single, simple access to the Internet. Initially, the topology only have one connection. In such case, BGP would not be really needed because a default gateway would suffice. However, as the explanation advances, elements will be added to this initial topology that will make the point for having BGP. This is the starting point topology and its general description.

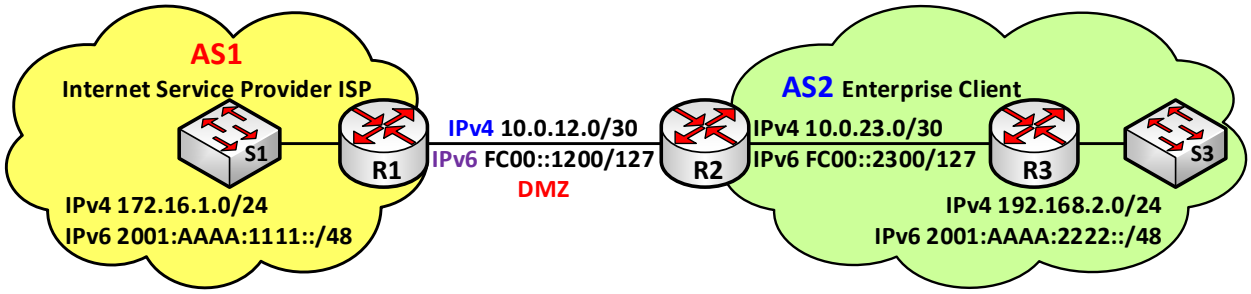


Figure 9.1

The Internet Service Provider AS1 grants access to the rest of Internet via a single link to the client AS2. Both network are dual stacked IPv4 and IPv6. It is assumed that AS1 owns the IPv6 address space 2001:AAAA::/32 and from there it is leasing the block 2001:AAAA:2222::/48 to AS2. The rest of the IPv6 addresses used for the DMZ links come from the private space FC00::/16. The configuration of all routers is done with both Cisco and Juniper commands and that is included in this analysis.

The general idea is to have the ISP providing a default route only to the customer as this figure illustrates, a BGP UPDATE will carry such information. Once installed in R2, the default route is

to be redistributed into the IGP running inside AS2. Basically, all the internal routers get to know that there is one way out of the domain via R2. The advantage of this approach is that the default information is dynamically generated by the service provider. If the link between R1 and R2 went down, then the default entry disappears which makes sense.

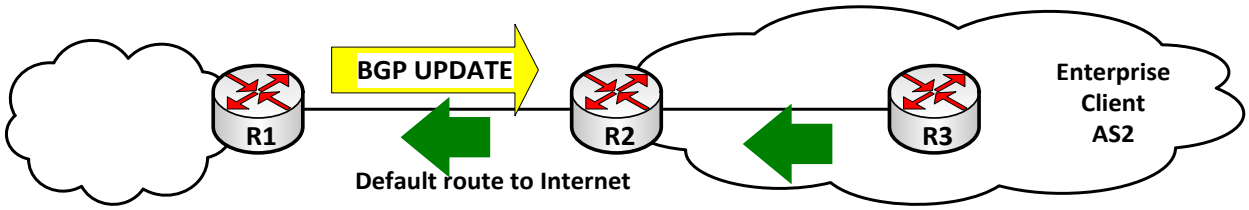


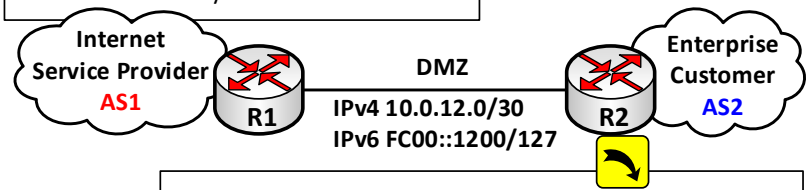
Figure 9.2

The first configuration is done with Cisco IOS. The key commands to be observed are "neighbor 10.12.12.2 default-originate" and "neighbor FC00::1201 default-originate."

```
hostname R1

router bgp 1
 neighbor 10.0.12.2 remote-as 2
 neighbor FC00::1201 remote-as 2
 !
 address-family ipv4
  neighbor 10.0.12.2 activate
  neighbor 10.0.12.2 default-originate
 exit-address-family
 !
 address-family ipv6
  neighbor FC00::1201 activate
  neighbor FC00::1201 default-originate
 exit-address-family
```

Originates a default IPv4 route 0.0.0.0/0 to be sent to neighbour R2

Originates a default IPv6 route ::/0 to be sent to neighbour R2



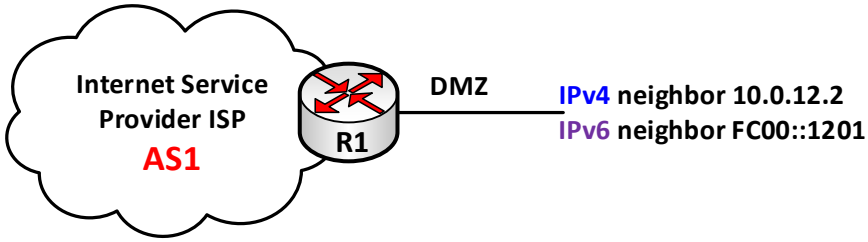

```
hostname R2
!
router bgp 2
 neighbor 10.0.12.1 remote-as 1
 neighbor FC00::1200 remote-as 1
 !
 address-family ipv4
  neighbor 10.0.12.1 activate
  network 192.168.2.0
 exit-address-family
 !
 address-family ipv6
  neighbor FC00::1200 activate
  network 2001:AAAA:2222::/48
 exit-address-family
```

Results

```
R2#show ip route
Gateway of last resort is 10.0.12.1 to network 0.0.0.0
B*    0.0.0.0/0 [20/0] via 10.0.12.1
Plus all other IPv4 Internet routes (over 600K)

R2#show ipv6 route
B   ::/0 [20/0] via FE80::C801:1FFF:FE2C:0, f0/0
Plus all other IPv6 Internet routes (over 40K)
```

Figure 9.3

The Cisco command "default-originate" when applied to a BGP neighbour creates a default route 0.0.0.0/0 for IPv4 and ::/0 for IPv6 only for that specific peer. Notice that the routing table at R2 demonstrates that, in fact, the default routes are being received correctly. However, the full Internet routing table is being received as well and for a topology with just one way in and out, that is unneeded, so the ISP has to do some filtering to **advertise only** the default route.




```
hostname R1
!
router bgp 1
neighbor 10.0.12.2 remote-as 2
neighbor FC00::1201 remote-as 2
!
address-family ipv4 unicast
network 172.16.1.0 mask 255.255.255.0
neighbor 10.0.12.2 activate
neighbor 10.0.12.2 default-originate
neighbor 10.0.12.2 route-map IPv4_DEFAULT_ONLY out
exit-address-family
!
address-family ipv6 unicast
neighbor FC00::1201 activate
neighbor FC00::1201 default-originate
neighbor FC00::1201 route-map IPv6_DEFAULT_ONLY out
exit-address-family
```

```
!
ip prefix-list IPv4-DEFAULT seq 5 permit 0.0.0.0/0
!
route-map IPv4_DEFAULT_ONLY permit 10
match ip address prefix-list IPv4-DEFAULT
route-map IPv4_DEFAULT_ONLY deny 20
!
ipv6 prefix-list IPv6-DEFAULT seq 5 permit ::/0
!
route-map IPv6_DEFAULT_ONLY permit 10
match ip address prefix-list IPv6-DEFAULT
route-map IPv6_DEFAULT_ONLY deny 20
!
```

Figure 9.4

The results are observed in router R2.

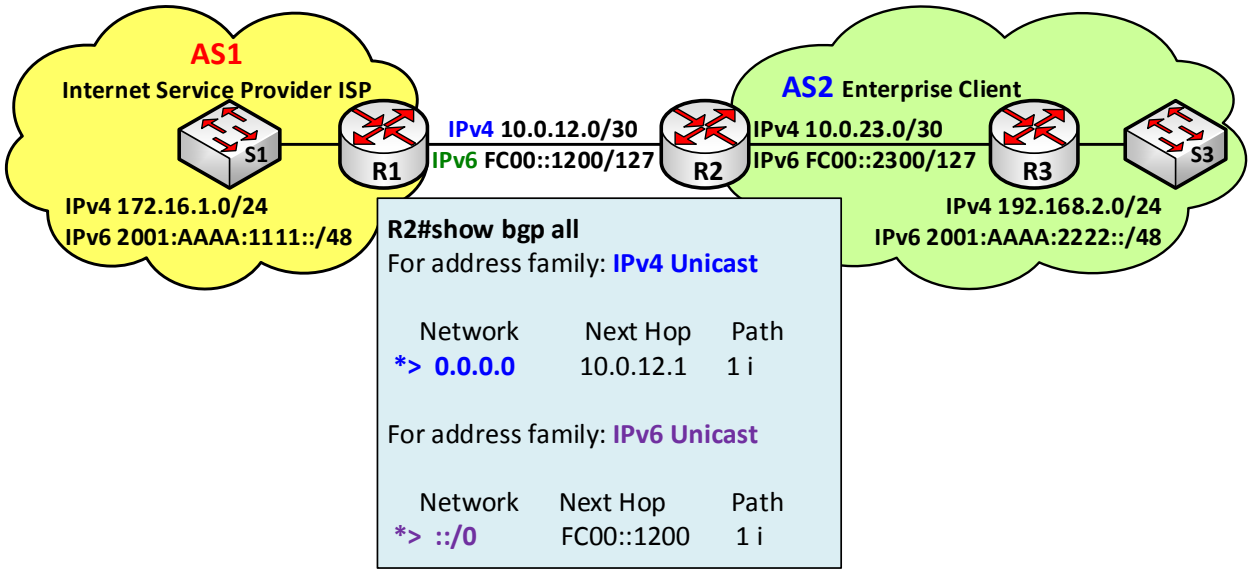


Figure 9.5

Now R2 has the corresponding default routes dynamically introduced via BGP. These default routes can be redistributed into and IGP so the routers inside the domain, like R3, can use to forward traffic toward the Internet.

It is the turn to repeat the configuration to show Juniper JUNOS way. This is the configuration file, step by step, for J1 in AS1.

```
set system hostname J1
set interfaces em0 unit 0 family inet address 10.0.12.1/30
set interfaces em0 unit 0 family inet6 address fc00::1200/127
set interfaces lo0 unit 0 family inet address 10.1.1.1/32

set routing-options router-id 10.1.1.1
set routing-options autonomous-system 1

set routing-options static route 0.0.0.0/0 discard
set routing-options rib inet6.0 static route ::/0 discard
```

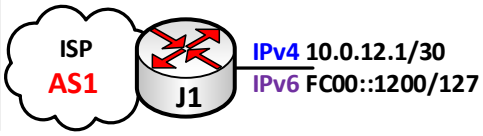


Figure 9.6

The default routes IPv4 0.0.0.0/0 and IPv6 ::/0 are statically created in J1 under the routing-options. Since these routes are meant to be advertised only to the customer AS2, they are configured as "discard" which means that J1 is not using the routes itself.

Next, two BGP groups are created for the customer AS2, one for IPv4 and another for IPv6. For each address family there is an export policy that matches the corresponding default route only. Anything else is rejected. In this manner, J1 advertises only 0.0.0.0/0 and ::/0 to J2.

```
set protocols bgp group IPv4_to_AS2 type external
set protocols bgp group IPv4_to_AS2 export DEFAULT_IPv4
set protocols bgp group IPv4_to_AS2 neighbor 10.0.12.2 peer-as 2

set protocols bgp group IPv6_to_AS2 type external
set protocols bgp group IPv6_to_AS2 family inet6 unicast
set protocols bgp group IPv6_to_AS2 export DEFAULT_IPv6
set protocols bgp group IPv6_to_AS2 neighbor fc00::1201 peer-as 2

set policy-options policy-statement DEFAULT_IPv4 term 1 from route-filter 0.0.0.0/0 exact
set policy-options policy-statement DEFAULT_IPv4 term 1 then accept
set policy-options policy-statement DEFAULT_IPv4 term 2 then reject

set policy-options policy-statement DEFAULT_IPv6 term 1 from route-filter ::/0 exact
set policy-options policy-statement DEFAULT_IPv6 term 1 then accept
set policy-options policy-statement DEFAULT_IPv6 term 2 then reject
```

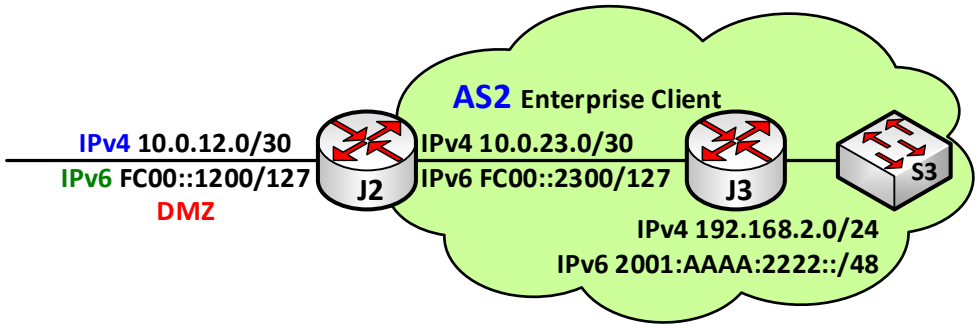
Additionally, the ISP has an inbound policy to accept only the routes advertised by AS2. This is the general idea using a regular expression to match only the sourcing AS2.

```
set protocols bgp group IPv4_to_AS2 type external
set protocols bgp group IPv4_to_AS2 neighbor 10.0.12.2 peer-as 2
set protocols bgp group IPv4_to_AS2 import INBOUND-from-AS2

set protocols bgp group IPv6_to_AS2 type external
set protocols bgp group IPv6_to_AS2 family inet6 unicast
set protocols bgp group IPv6_to_AS2 neighbor fc00::1201 peer-as 2
set protocols bgp group IPv6_to_AS2 import INBOUND-from-AS2

set policy-options policy-statement INBOUND-from-AS2 term 1 from as-path SOURCE-AS2
set policy-options policy-statement INBOUND-from-AS2 term 1 then accept
set policy-options policy-statement INBOUND-from-AS2 term 1 then reject

set policy-options as-path SOURCE-AS2  ".*2"
```

Although the previous policy works, this one below is a tighter, more granular version that matches the AS of origin and the exact IP address space. The best practice principle is not to take for granted that the other side is always going to advertise the correct prefixes.

```
set protocols bgp group IPv4_to_AS2 type external
set protocols bgp group IPv4_to_AS2 neighbor 10.0.12.2 peer-as 2
set protocols bgp group IPv4_to_AS2 import INBOUND-IPv4-from-AS2

set protocols bgp group IPv6_to_AS2 type external
set protocols bgp group IPv6_to_AS2 family inet6 unicast
set protocols bgp group IPv6_to_AS2 neighbor fc00::1201 peer-as 2
set protocols bgp group IPv6_to_AS2 import INBOUND-IPv6-from-AS2

set policy-options policy-statement INBOUND-IPv4-from-AS2 term 1 from as-path SOURCE-AS2
set policy-options policy-statement INBOUND-IPv4-from-AS2 term 1 from route-filter 192.168.2.0/24  exact
set policy-options policy-statement INBOUND-IPv4-from-AS2 then accept
set policy-options policy-statement INBOUND-IPv4-from-AS2 term 1 then reject

set policy-options policy-statement INBOUND-IPv6-from-AS2 term 1 from as-path SOURCE-AS2
set policy-options policy-statement INBOUND-IPv6-from-AS2 term 1 from route-filter 2001:AAAA:2222::/48 exact
set policy-options policy-statement INBOUND-IPv6-from-AS2 then accept
set policy-options policy-statement INBOUND-IPv6-from-AS2 term 1 then reject

set policy-options as-path SOURCE-AS2  ".*2"
```

Juniper router J2 configuration is displayed next in detail. First, the configuration of the interfaces and the IGP routing protocol IS-IS. This routing protocol distributes the default route information from BGP to the internal router J3. Hence the export command pointing to the policy "from_BGP_into_ISIS."




```
set system hostname J2
set interfaces lo0 unit 0 family inet address 10.2.2.2/32
set interfaces lo0 unit 0 family iso address 47.0000.bbbb.bbbb.bbbb.00
set interfaces em0 unit 0 family inet address 10.0.12.2/30
set interfaces em0 unit 0 family inet6 address fc00::1201/127
set interfaces em1 unit 0 family inet address 10.0.23.1/30
set interfaces em1 unit 0 family iso
set interfaces em1 unit 0 family inet6 address fc00::2300/127

set protocols isis interface em1.0
set protocols isis interface lo0.0
set protocols isis export from_BGP_into_ISIS

set policy-options policy-statement from_BGP_into_ISIS term 1 from protocol bgp
set policy-options policy-statement from_BGP_into_ISIS term 1 from route-filter 0.0.0.0/0 exact
set policy-options policy-statement from_BGP_into_ISIS term 1 then accept
set policy-options policy-statement from_BGP_into_ISIS term 2 from protocol bgp
set policy-options policy-statement from_BGP_into_ISIS term 2 from route-filter ::/0 exact
set policy-options policy-statement from_BGP_into_ISIS term 2 then accept
set policy-options policy-statement from_BGP_into_ISIS term 3 then reject
```

Figure 9.7

The BGP configuration of the customer side could be as simple as just defining the peer relationship and the export policy since the ISP is already implementing all the correct filters. Nevertheless, the administrator in the customer side should not take for granted that the other side is going to always advertise the correct information. So, the customer configuration could look like this for IPv4 routing:

```
set routing-options router-id 10.2.2.2
set routing-options autonomous-system 2

set protocols bgp group AS2_IPv4 type external
set protocols bgp group AS2_IPv4 import from_AS1_IPv4
set protocols bgp group AS2_IPv4 export to_ISP-AS1_IPv4
set protocols bgp group AS2_IPv4 neighbor 10.0.12.1 peer-as 1

set policy-options policy-statement from_AS1_IPv4 term 1 from protocol bgp
set policy-options policy-statement from_AS1_IPv4 term 1 from route-filter 0.0.0.0/0 exact
set policy-options policy-statement from_AS1_IPv4 term 1 then accept
set policy-options policy-statement from_AS1_IPv4 term 2 then reject
```

The policies for importing IPv4 and IPv6 routes can be these:

```
set protocols bgp group AS2_IPv6 type external
set protocols bgp group AS2_IPv6 neighbor fc00::1200 family inet6 unicast
set protocols bgp group AS2_IPv6 neighbor fc00::1200 peer-as 1
set protocols bgp group AS2_IPv6 import from_AS1_IPv6
set protocols bgp group AS2_IPv6 export to_ISP-AS1_IPv6

set policy-options policy-statement from_AS1_IPv6 term 1 from protocol bgp
set policy-options policy-statement from_AS1_IPv6 term 1 from route-filter ::/0 exact
set policy-options policy-statement from_AS1_IPv6 term 1 then accept
set policy-options policy-statement from_AS1_IPv6 term 2 then reject

set policy-options policy-statement to_ISP-AS1_IPv6 term 1 from route-filter 2001:AAAA:2222::/48 exact
set policy-options policy-statement to_ISP-AS1_IPv6 term 1 then accept
set policy-options policy-statement to_ISP-AS1_IPv6 term 2 then reject
```

The last router (J3) inside AS2 has no BGP at all but it is getting the default routes dynamically via the IGP IS-IS.

```
set system hostname J3

set interfaces lo0 unit 0 family inet address 10.3.3.3/32
set interfaces lo0 unit 0 family iso address 47.0000.cccc.cccc.cccc.00
set interfaces em0 unit 0 family inet address 192.168.2.1/24
set interfaces em0 unit 0 family inet6 address 2001:AAAA:2222::1/48
set interfaces em0 unit 0 family iso
set interfaces em1 unit 0 family inet address 10.0.23.2/30
set interfaces em1 unit 0 family inet6 address fc00::2301/127
set interfaces em1 unit 0 family iso

set protocols isis interface all
```

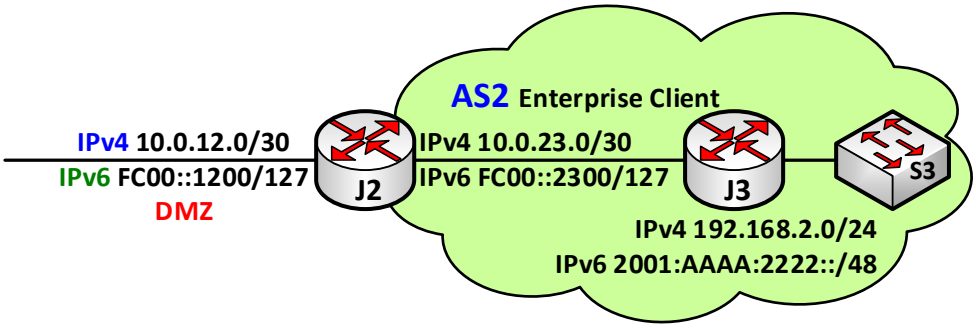


Figure 9.8

```
J3# show route terse (edited for simplicity)
inet.0                    Next_Hop
0.0.0.0/0                 10.0.23.1
inet6.0
::/0                      FC00::2300
(Plus internal and directly connected routes)
```

Thus, the configuration has achieved the simple access to Internet via default routes sourced by an ISP. The previous topology is modified now to add a new scenario.

Two AS peering to exchange local routes

In the previous configuration, AS1 provides access to the Internet by supplying default routes only to AS2. In the meantime, AS2 advertises its whole address space to AS1. In such arrangement, AS1 can further propagate this routing information to its own peers. Consequently, the rest of Internet gets to know how to get to AS2. But from the viewpoint of AS2, the Internet is just a couple of routes: IPv4 0.0.0.0/0 and IPv6 ::/0. This is simple and it works.

However, the administration in AS2 also wants to peer directly to another, similarly set up, AS3. There are many reasons why this would need to be done, but in general, it suffices to say that there is an interest for direct traffic between the two organizations. The two AS could contract a physical link, such as a dark optical fibre, between the two sites or they could meet at an exchange point which is the case represented in the next diagram. **Internet Exchange Points (IXP)** [1] are actually buildings with the infrastructure to make the interconnections. There are many IXP locations all over the world. In the current example, both AS2 and AS3 have presence in the same exchange and the two domains become physically linked as shown.

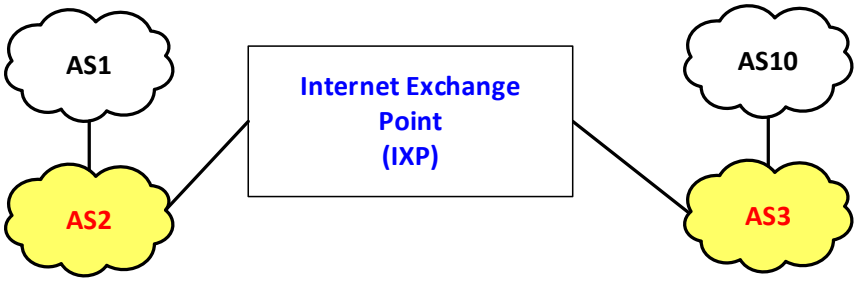


Figure 9.9

The next step is to declare the routing intentions of each AS. In this example, the BGP peering arrangement is to reach each other local address spaces, or a subset of those, and nothing else. That means that AS3 cannot reach AS1 via AS2 and vice versa. In this case, the next pseudo-code policy illustrate the point.

```
To neighbor AS:
        Send only local routes
        (this can be done matching the AS number)
        Send particular prefixes
        (this can be done matching with a route filter
        or a combination of matching AS and route filtering.
```

A specific topology is presented in the following diagram and the configuration is done with Cisco IOS. It is wise to arrange the peers in different profiles by using peer-groups to keep them more organized. Also, peer groups ease the outgoing update load [2] of a Cisco router. This is the configuration for R2-AS2 and only the policies toward AS3 are being shown. The policies toward AS1 are being kept from the previous example.

```
router bgp 2

neighbor AS1-IPv4 peer-group
neighbor AS1-IPv4 description "Upstream IPv4 to AS1 Internet"
neighbor AS1-IPv4 remote-as 1
neighbor 10.0.12.1 peer-group AS1-IPv4

neighbor AS1-IPv6 peer-group
neighbor AS1-IPv6 description "Upstream IPv6 to AS1 Internet"
neighbor AS1-IPv6 remote-as 1
neighbor FC00::1200 peer-group AS1-IPv6

neighbor AS3-IPv4 peer-group
neighbor AS3-IPv4 description "Peer AS3 only local IPv4 routes"
neighbor AS3-IPv4 remote-as 3
neighbor 10.0.23.2 peer-group AS3-IPv4

neighbor AS3-IPv6 peer-group
neighbor AS3-IPv6 description "Peer AS3 only local IPv6 routes"
neighbor AS3-IPv6 remote-as 3
neighbor FC00::2301 peer-group AS3-IPv6
```

```
Continuation …

address-family ipv4 unicast
neighbor 10.0.12.1 activate
neighbor 10.0.23.2 activate
exit-address-family
!
address-family ipv6 unicast
neighbor FC00::1200 activate
neighbor FC00::2301 activate
exit-address-family
```

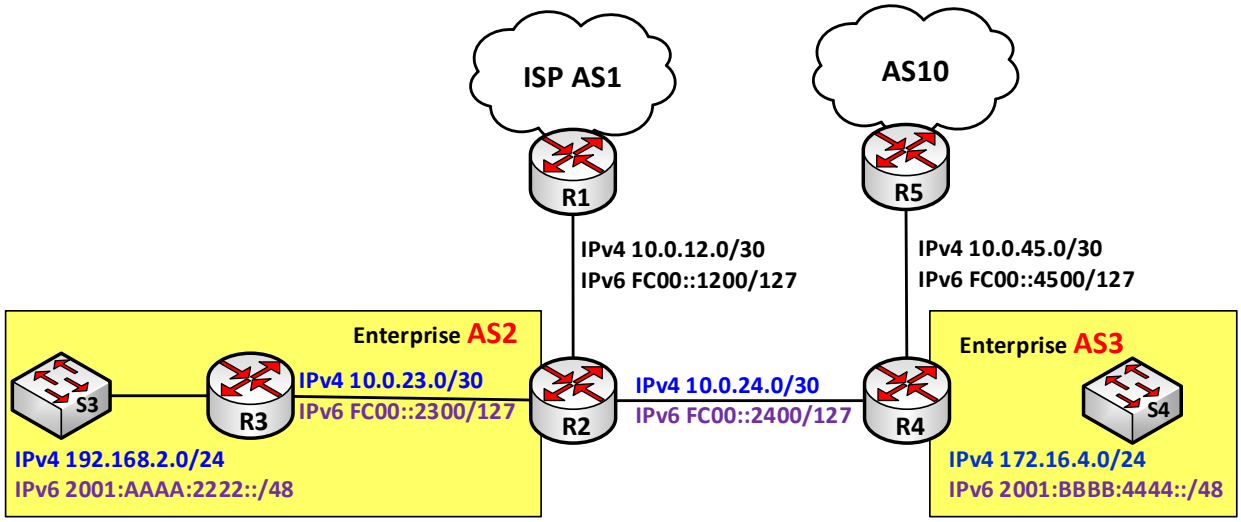


Figure 9.10

The outbound policy could be as simple as matching the local AS only which is done with the "empty path" regular expression **^$**.

```
address-family ipv4 unicast
neighbor 10.0.12.1 activate
neighbor 10.0.23.2 activate
neighbor AS3-IPv4 route-map EXPORT-LOCAL out
exit-address-family
!
address-family ipv6 unicast
neighbor FC00::1200 activate
neighbor FC00::2301 activate
neighbor AS3-IPv6 route-map EXPORT-LOCAL out
exit-address-family
```

```
ip as-path access-list 2 permit ^$
!
route-map EXPORT-LOCAL permit 10
 match as-path 2
!
route-map EXPORT-LOCAL deny 20
```

An inbound policy to accept only the local routes sourced from AS3 is also needed. Since Cisco peer groups are meant only for outbound updates, the inbound policies have to be applied specifically to the neighbour IP destination. The regular expression ^3$ matches exactly the routes sourced by a directly connected AS3.

```
address-family ipv4 unicast
neighbor 10.0.12.1 activate
neighbor 10.0.23.2 activate
neighbor AS3-IPv4 route-map EXPORT-LOCAL out
neighbor 10.0.23.2 route-map INBOUND-from-AS3 in
exit-address-family
!
address-family ipv6 unicast
neighbor FC00::1200 activate
neighbor FC00::2301 activate
neighbor FC00::2301 route-map INBOUND-from-AS3 in
neighbor AS3-IPv6 route-map EXPORT-LOCAL out
exit-address-family
```

```
ip as-path access-list 3 permit ^3$
!
route-map INBOUND-from-AS3 permit 10
 match as-path 3
!
route-map INBOUND-from-AS3 deny 20
```

Multiple paths between two autonomous systems

A client AS might have several links to the same provider to obtain redundancy. In such case, the objective is to make an efficient utilization of the physical resources. The following network topology is used to explore several peering techniques. (Disclaim: this topology is purposely designed to illustrate the techniques and it is not a best practice design. In fact, having just one border router is a liability.)

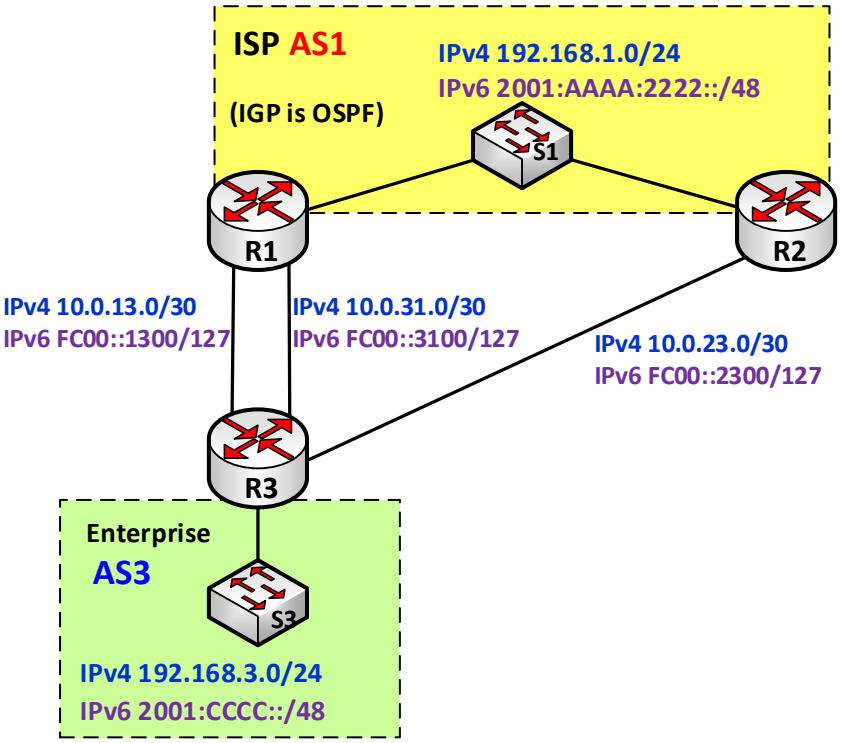
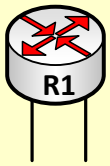


Figure 9.11

All the networks are dual stack IPv4 and IPv6. Two identical physical links are provisioned between R1 and R3 and another connection is placed between R2 and R3. Inside AS1, OSPF is being run for both IPv4 and IPv6. Routers R1 and R2 peer with iBGP using their loopback addresses 10.1.1.1 and 10.2.2.2. Beside that, R1 eBGP peers with R3 and R2 eBGP peers with R3. The next snippets contain the BGP configuration of the three routers.

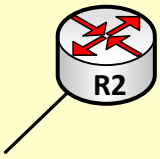
Cisco IOS Implementation

hostname **R1**



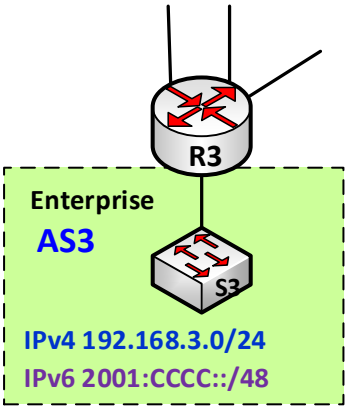

```
router bgp 1
 neighbor INTERNAL-IPv4 peer-group
 neighbor INTERNAL-IPv4 remote-as 1
 neighbor INTERNAL-IPv4 update-source Loopback0
 neighbor INTERNAL-IPv6 peer-group
 neighbor INTERNAL-IPv6 remote-as 1
 neighbor AS3-IPv4 peer-group
 neighbor AS3-IPv4 remote-as 3
 neighbor AS3-IPv6 peer-group
 neighbor AS3-IPv6 remote-as 3
 neighbor 10.2.2.2 peer-group INTERNAL-IPv4
 neighbor 2001:AAAA::2 peer-group INTERNAL-IPv6
 neighbor 10.0.13.2 peer-group AS3-IPv4
 neighbor 10.0.31.2 peer-group AS3-IPv4
 neighbor FC00::1301 peer-group AS3-IPv6
 neighbor FC00::3101 peer-group AS3-IPv6
 !
 address-family ipv4 unicast
  network 192.168.1.0
  neighbor INTERNAL-IPv4 next-hop-self
  neighbor 10.2.2.2 activate
  neighbor 10.0.13.2 activate
  neighbor 10.0.31.2 activate
 exit-address-family
 !
 address-family ipv6 unicast
  network 2001:AAAA::/48
  neighbor INTERNAL-IPv6 next-hop-self
  neighbor 2001:AAAA::2 activate
  neighbor FC00::1301 activate
  neighbor FC00::3101 activate
 exit-address-family
!
```

hostname **R2**




```
router bgp 1
 neighbor INTERNAL-IPv4 peer-group
 neighbor INTERNAL-IPv4 remote-as 1
 neighbor INTERNAL-IPv4 update-source Loopback0
 neighbor INTERNAL-IPv6 peer-group
 neighbor INTERNAL-IPv6 remote-as 1
 neighbor AS3-IPv4 peer-group
 neighbor AS3-IPv4 remote-as 3
 neighbor AS3-IPv6 peer-group
 neighbor AS3-IPv6 remote-as 3
 neighbor 10.1.1.1 peer-group INTERNAL-IPv4
 neighbor 2001:AAAA::1 peer-group INTERNAL-IPv6
 neighbor 10.0.23.2 peer-group AS3-IPv4
 neighbor FC00::2301 peer-group AS3-IPv6
!
address-family ipv4 unicast
 network 192.168.1.0
 neighbor INTERNAL-IPv4 next-hop-self
 neighbor 10.1.1.1 activate
 neighbor 10.0.23.2 activate
exit-address-family
 !
address-family ipv6 unicast
 network 2001:AAAA::/120
 neighbor INTERNAL-IPv6 next-hop-self
 neighbor 2001:AAAA::1 activate
 neighbor FC00::2301 activate
exit-address-family
```

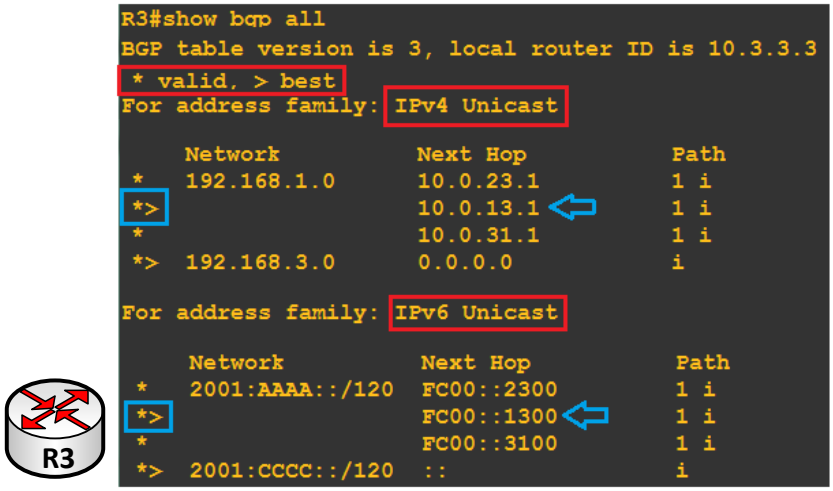
Figure 9.12




```
hostname R3
router bgp 3
 neighbor 10.0.13.1 remote-as 1
 neighbor 10.0.23.1 remote-as 1
 neighbor 10.0.31.1 remote-as 1
 neighbor FC00::1300 remote-as 1
 neighbor FC00::3100 remote-as 1
 neighbor FC00::2300 remote-as 1
 !
 address-family ipv4 unicast
  network 192.168.3.0
  neighbor 10.0.13.1 activate
  neighbor 10.0.23.1 activate
  neighbor 10.0.31.1 activate
exit-address-family
 !
 address-family ipv6 unicast
  network 2001:CCCC::/120
  neighbor FC00::1300 activate
  neighbor FC00::3100 activate
  neighbor FC00::2300 activate
exit-address-family
```

Figure 9.13

This is BGP viewpoint from R3, once that the routing information has converged.




```
R3#show bgp all
BGP table version is 3, local router ID is 10.3.3.3
 * valid, > best
For address family: IPv4 Unicast

     Network          Next Hop            Path
 *   192.168.1.0      10.0.23.1           1 i
 *>                   10.0.13.1           1 i
 *                    10.0.31.1           1 i
 *>  192.168.3.0      0.0.0.0             i

For address family: IPv6 Unicast

     Network           Next Hop            Path
 *   2001:AAAA::/120   FC00::2300          1 i
 *>                    FC00::1300          1 i
 *                     FC00::3100          1 i
 *>  2001:CCCC::/120   ::                  i
```

Figure 9.14

For every destination prefix, IPv4 192.168.1.0/24 and IPv6 2001:AAAA::/120, there are three possible paths. For each destination, only one path is chosen to be the best as indicated by the arrows in the diagram. For IPv4, the best path is the one with the lowest address 10.0.13.1 and for IPv6, the best path is the lowest address FC00::1300. Furthermore, the IPv4 routing table proves that only the best path gets installed for forwarding packets.

```
R3#show ip route
B     192.168.1.0/24 [20/0] via 10.0.13.1 FastEthernet1/0
```

Also, the IPv6 routing table shows the same behaviour.

```
R3#show ipv6 route
B   2001:AAAA::/120 [20/0]
     via FE80::C801:2DFF:FECC:1C, FastEthernet1/0
```

This is all correct according to the BGP default process. The basic rule is that BGP chooses only one path out of multiple paths to the same destination. The selection process considers several attributes first but if a tie results, BGP selects the path with the next-hop lowest IP address which is exactly what has happened in this example.

A consequence of this BGP rule, in this particular topology, is that the other links are sitting idle. But if the best path were to fail, the next best will be promoted to the routing table. There is nothing wrong with that. In fact, that is the way that BGP operates if left alone. BGP does not do load-balancing by default. Nevertheless, the additional links represents a fixed cost that should have a better utilization, especially if the only link in operation is overloaded. To enable all the links to operate at the same time, Cisco IOS has the command "**maximum-paths**."

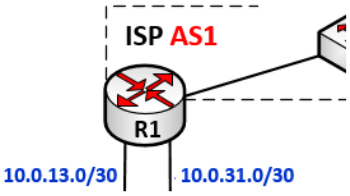
```
router bgp 3
 neighbor 10.0.13.1 remote-as 1
 neighbor 10.0.23.1 remote-as 1
 neighbor 10.0.31.1 remote-as 1
 neighbor FC00::1300 remote-as 1
 neighbor FC00::2300 remote-as 1
 neighbor FC00::3100 remote-as 1
 !
 address-family ipv4
  maximum-paths 3
  network 192.168.3.0
  neighbor 10.0.13.1 activate
  neighbor 10.0.23.1 activate
  neighbor 10.0.31.1 activate
 exit-address-family
 !
 address-family ipv6
  maximum-paths 3
  network 2001:CCCC::/120
  neighbor FC00::1300 activate
  neighbor FC00::2300 activate
  neighbor FC00::3100 activate
 exit-address-family
```

```
R3#show ip route bgp

B     192.168.1.0/24 [20/0] via 10.0.31.1 FastEthernet2/0
                     [20/0] via 10.0.23.1 FastEthernet1/1
                     [20/0] via 10.0.13.1 FastEthernet1/0
```

```
R3#show ipv6 route bgp
IPv6 Routing Table
B   2001:AAAA::/120 [20/0]
     via FE80::C801:2DFF:FECC:1C, FastEthernet1/0
     via FE80::C803:25FF:FE48:1D, FastEthernet1/1
     via FE80::C801:2DFF:FECC:38, FastEthernet2/0
```

Figure 9.14

The previous snippet shows the effect of the "maximum-path" command under the respective address families. In AS1, R1 is the only router that needs to have multiple paths enabled. The letter m is the indicative of the BGP multipath entry.




```
R1#show ip bgp
BGP table version is 3, local router ID is 10.1.1.1
     Network          Next Hop            Metric LocPrf Weight Path
 * i 192.168.3.0      10.2.2.2  (from iBGP)     0    100      0 3 i
 *m (m for multipath) 10.0.31.2 (external links) 0             0 3 i
 *> (best path)       10.0.13.2                 0             0 3 i
```

Figure 9.15

In router R3, the IPv4 routing table has the prefix 192.168.1.0/24 with three different equal paths. In principle, load balancing can be tested with traceroute tools

```
R3# traceroute 192.168.1.1 source 192.168.3.1

10.0.13.1
10.0.23.1   load balancing
10.0.31.1
```

```
R3# traceroute
Protocol: ipv6
Target: 2001:AAAA::1
Source: 2001:CCCC::1

FC00::1300
FC00::3100   load balancing
FC00::2300
```

It works. Each traceroute message is sent to a different interface in a round-robin basis.

Junos Implementation

This simple topology of two Juniper routers will be used to demonstrate the configuration of multiple paths between two autonomous systems.

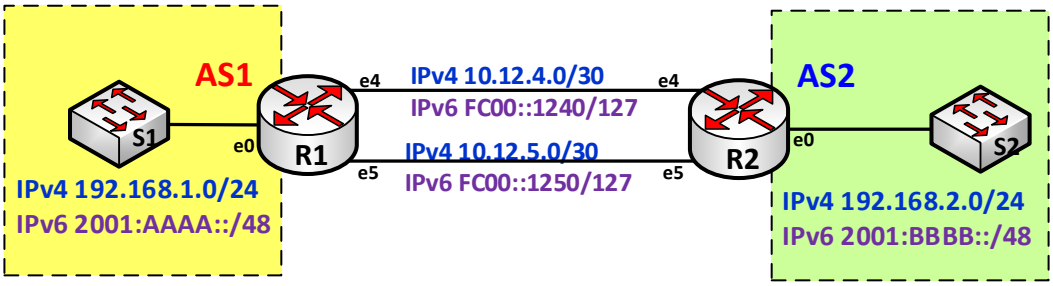


Figure 9.16

These are the interface configurations for both routers is displayed side by side.

root@J1> show configuration | display set set system host-name J1 set interfaces em0.0 family inet address 192.168.1.1/24 set interfaces em0.0 family inet6 address 2001:AAAA::1/48 set interfaces em4.0 family inet address 10.12.4.1/30 set interfaces em4.0 family inet6 address fc00::1240/127 set interfaces em5.0 family inet address 10.12.5.1/30 set interfaces em5.0 family inet6 address fc00::1250/127 set interfaces lo0.0 family inet address 10.1.1.1/32	root@J2> show configuration | display set set system host-name J2 set interfaces em0.0 family inet address 192.168.2.1/24 set interfaces em0.0 family inet6 address 2001:BBBB::1/48 set interfaces em4.0 family inet address 10.12.4.2/30 set interfaces em4.0 family inet6 address fc00::1241/127 set interfaces em5.0 family inet address 10.12.5.2/30 set interfaces em5.0 family inet6 address fc00::1251/127 set interfaces lo0.0 family inet address 10.2.2.2/32

There are three actions to activate multipath with load balancing in Junos. Multipath must be activated under protocol BGP, but that alone does not implement load balancing. Instead, a policy that matches certain condition, like a prefix or an AS path, must be created and the action to load balance per packet must be specified. Then, the output of this policy is installed, via an export command, into the forwarding-table of the router. This snippet contains the basic idea.

```
set routing-options forwarding-table export LOAD-BALANCING

set policy-options policy-statement LOAD-BALANCING term 1 from PLACE-CONDITION-HERE
set policy-options policy-statement LOAD-BALANCING term 1 then load-balance per-packet
set policy-options policy-statement LOAD-BALANCING term 1 then accept

set protocols bgp group EXAMPLE multipath
```

Consequently, this is the rest of the configuration for router R1 (or J1).

```
root@J1> show configuration | display set
set routing-options router-id 10.1.1.1
set routing-options autonomous-system 1
set routing-options forwarding-table export LOAD-BALANCING

set protocols bgp group AS1-IPv4 type external
set protocols bgp group AS1-IPv4 multipath
set protocols bgp group AS1-IPv4 import IMPORT-IPv4-from-AS2
set protocols bgp group AS1-IPv4 export LOCAL-IPv4
set protocols bgp group AS1-IPv4 neighbor 10.12.4.2 peer-as 2
set protocols bgp group AS1-IPv4 neighbor 10.12.5.2 peer-as 2

set protocols bgp group AS1-IPv6 type external
set protocols bgp group AS1-IPv6 family inet6 unicast
set protocols bgp group AS1-IPv6 multipath
set protocols bgp group AS1-IPv6 import IMPORT-IPv6-from-AS2
set protocols bgp group AS1-IPv6 export LOCAL-IPv6
set protocols bgp group AS1-IPv6 neighbor fc00::1241 peer-as 2
set protocols bgp group AS1-IPv6 neighbor fc00::1251 peer-as 2

set policy-options policy-statement LOAD-BALANCING term 1 from as-path AS2
set policy-options policy-statement LOAD-BALANCING term 1 then load-balance per-packet
set policy-options policy-statement LOAD-BALANCING term 1 then accept

set policy-options as-path AS2 "^2$"

set policy-options policy-statement LOCAL-IPv4 term 1 from route-filter 192.168.1.0/24 exact
set policy-options policy-statement LOCAL-IPv4 term 1 then accept
set policy-options policy-statement LOCAL-IPv4 term 2 then reject

set policy-options policy-statement LOCAL-IPv6 term 1 from route-filter 2001:AAAA::/48 exact
set policy-options policy-statement LOCAL-IPv6 term 1 then accept
set policy-options policy-statement LOCAL-IPv6 term 2 then reject

set policy-options policy-statement IMPORT-IPv4-from-AS2 term 1 from route-filter 192.168.2.0/24 exact
set policy-options policy-statement IMPORT-IPv4-from-AS2 term 1 then accept
set policy-options policy-statement IMPORT-IPv4-from-AS2 term 2 then reject

set policy-options policy-statement IMPORT-IPv6-from-AS2 term 1 from route-filter fc00:BBBB::/48 exact
set policy-options policy-statement IMPORT-IPv6-from-AS2 term 1 then accept
set policy-options policy-statement IMPORT-IPv6-from-AS2 term 2 then reject
```

The other router R2 has a mirror configuration with respect this one.

```
set system host-name J2
set interfaces em0 unit 0 family inet address 192.168.2.1/24
set interfaces em0 unit 0 family inet6 address 2001:BBBB::1/48
set interfaces em4 unit 0 family inet address 10.12.4.2/30
set interfaces em4 unit 0 family inet6 address fc00::1241/127
set interfaces em5 unit 0 family inet address 10.12.5.2/30
set interfaces em5 unit 0 family inet6 address fc00::1251/127
set interfaces lo0 unit 0 family inet address 10.2.2.2/32
set routing-options rib inet6.0 maximum-paths 3
set routing-options router-id 10.2.2.2
set routing-options autonomous-system 2
set routing-options forwarding-table export LOAD-BALANCE-AS2
set protocols bgp group AS2-IPv4 type external
set protocols bgp group AS2-IPv4 export LOCAL-IPv4
set protocols bgp group AS2-IPv4 multipath
set protocols bgp group AS2-IPv4 neighbor 10.12.4.1 peer-as 1
set protocols bgp group AS2-IPv4 neighbor 10.12.5.1 peer-as 1
set protocols bgp group AS2-IPv6 type external
set protocols bgp group AS2-IPv6 family inet6 unicast
set protocols bgp group AS2-IPv6 export LOCAL-IPv6
set protocols bgp group AS2-IPv6 multipath
set protocols bgp group AS2-IPv6 neighbor fc00::1250 peer-as 1
set protocols bgp group AS2-IPv6 neighbor fc00::1240 peer-as 1
set policy-options policy-statement LOAD-BALANCE-AS2 term 1 from protocol bgp
set policy-options policy-statement LOAD-BALANCE-AS2 term 1 from as-path AS1
set policy-options policy-statement LOAD-BALANCE-AS2 term 1 then load-balance per-packet
set policy-options policy-statement LOAD-BALANCE-AS2 term 1 then accept
set policy-options policy-statement LOCAL-IPv4 term 1 from route-filter 192.168.2.0/24 exact
set policy-options policy-statement LOCAL-IPv4 term 1 then accept
set policy-options policy-statement LOCAL-IPv6 term 1 from route-filter 2001:BBBB::/48 exact
set policy-options policy-statement LOCAL-IPv6 term 1 then accept
set policy-options policy-statement LOCAL-IPv6 term 2 then reject
set policy-options as-path AS1 "^1$"
```

Notice that the LOAD-BALANCING policy matches the routes sourced by AS2 only and that is inclusive for both IPv4 and IPv6. The specific prefixes are filtered by the importing policies. Then whatever passes thru both policies is installed in in the forwarding-table.

This is the result for the IPv4 prefix 192.168.2.0/24 located in AS2:

```
root@J1> show route forwarding-table destination 192.168.2.0/24
Routing table: default.inet
Internet:
Destination                       Next hop                    Netif
192.168.2.0/24                                                2
                                  10.12.4.2                    3 em4.0
                                  10.12.5.2                    4 em5.0
```

While this is the result for IPv6 2001:BBBB::/48, also located in AS2:

```
root@J1> show route forwarding-table destination 2001:BBBB::/48
Routing table: default.inet6
Internet6:
Destination                       Next hop                    Netif
2001:bbbb::/48                                                2
                                  fc00::1241                   3 em4.0
                                  fc00::1251                   4 em5.0
```

It looks like these prefixes have two possible paths, one via interface em4.0 and the other via interface em5.0. A combination of pinging and sniffing traffic validates the functioning of the policy. The IPv4 packets are destined to 192.168.2.1 and the diagram includes the MAC addresses of the interfaces to make the point that two different physical links are being used.

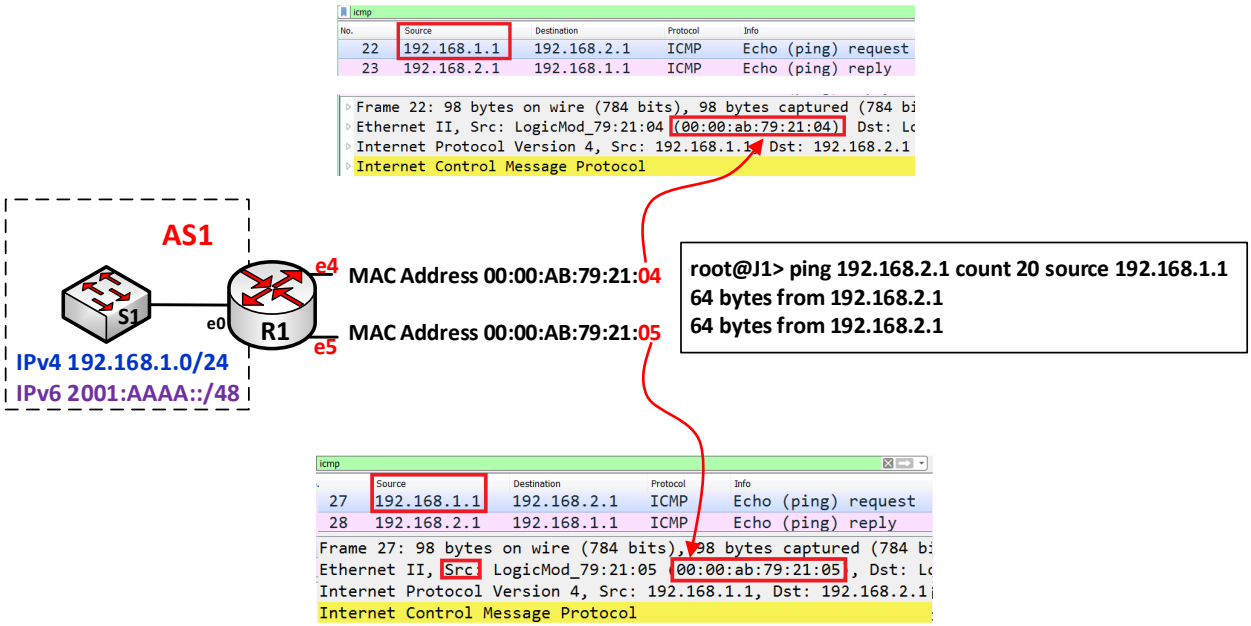


Figure 9.17

The same test is conducted for the IPv6 address 2001:BBBB::1 located in AS2.

```
root@J1> ping 2001:BBBB::1 count 20 source 2001:AAAA::1
PING6(56=40+8+8 bytes) 2001:aaaa::1 --> 2001:bbbb::1
16 bytes from 2001:bbbb::1, icmp_seq=0 hlim=64 time=56.630 ms
16 bytes from 2001:bbbb::1, icmp_seq=1 hlim=64 time=58.732 ms
16 bytes from 2001:bbbb::1, icmp_seq=2 hlim=64 time=154.773 ms
16 bytes from 2001:bbbb::1, icmp_seq=3 hlim=64 time=44.640 ms
16 bytes from 2001:bbbb::1, icmp_seq=4 hlim=64 time=71.749 ms
16 bytes from 2001:bbbb::1, icmp_seq=5 hlim=64 time=35.978 ms
```

The IPv6 packets are sent alternated between the two interfaces as the different MAC addresses attest in the following snippets. The first IPv6 packet going to 2001:BBBB::1 is encapsulated by Ethernet with source MAC address 00:00:AB:79:21:04, which is one of the interfaces of R1.

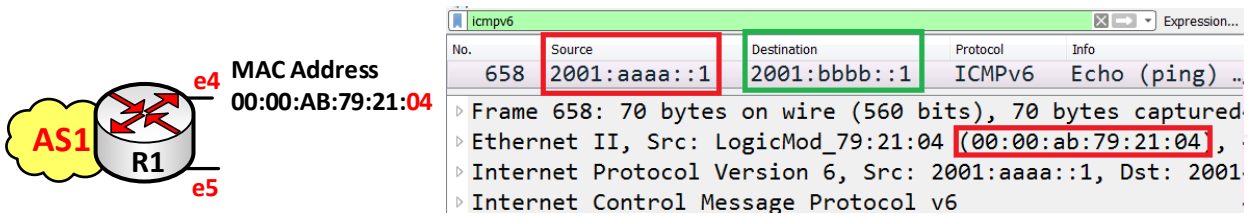


Figure 9.18

The second IPv6 packet going to 2001:BBBB::1 is encapsulated by Ethernet with a source MAC address 00:00:AB:79:21:05, which is the other physical interfaces of R1.

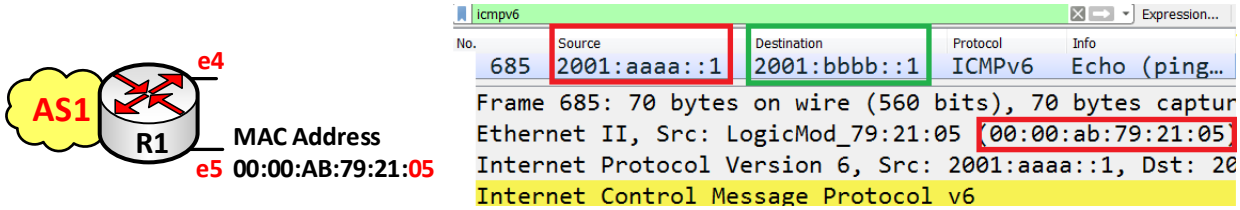


Figure 9.19

This concludes the configuration and explanation of load balancing both IPv4 and IPv6 traffic with a BGP configuration.

Chapter Summary

The most common BGP requirement is the configuration of a customer to access Internet via service providers ISPs. There are several modalities in the implementation of such service. Sometimes, the customer only requires very basic access and the ISP only need to supply default route information.

On the other hand, the client might have several links (multilink) to either the same provider or to different providers (multihome). In these situations, BGP by default algorithm chooses one best path. However, the client's intention is to make use of all the physical links as optimally as possible. Consequently, the provisioning becomes more complex because policies are needed to enable multipath usage with load balancing. These situations were described in this chapter for both IPv4 and IPv6 and the configuration was done with both Cisco IOS and Juniper JUNOS.

Chapter References

[1] BGP Best Path Selection Algorithm. Document ID: 13753. April 2012. www.cisco.net

[2] Configuring the Preference Value for BGP Routes. (2017). Juniper Networks TechLibrary.

http://www.juniper.net/documentation/en_US/junos11.4/topics/example/bgp-preference.html

[3] Toronto Internet Exchange TorIX. (2016) https://www.torix.ca

[4] Randy Zhang. Micah Bartell. (2004). BGP Design and Implementation. Cisco Press.

CHAPTER 10

BGP Multihome Organization

This chapter focuses on the use of BGP techniques to control the access to the Internet by an enterprise network. Several policies are implemented to match either BGP attributes or prefixes to steer the traffic in a network that resembles a real life situation.

Network topology

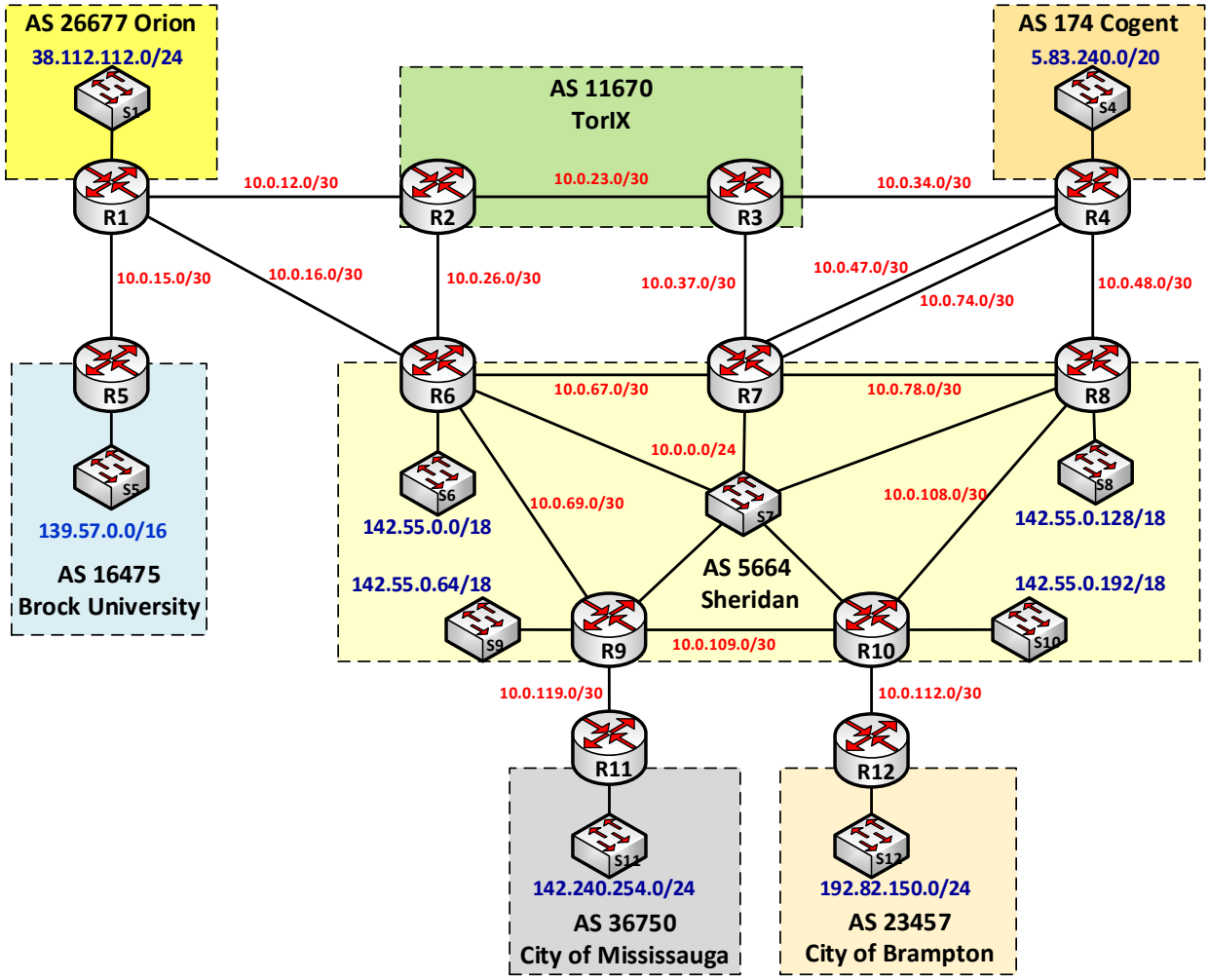


Figure 10.1

Disclaimer: this is a **pretend network for educational purposes only** created from information openly available in the public domain. The names, AS numbers and prefixes are real but everything else is just made up. This is the network topology for this case study.

Topology general description

This made-up topology is centered on Sheridan College's routing needs since the central topic is how the enterprise network connects to the Internet. In real life, Sheridan holds [1] the autonomous system number 5664, the IPv4 address space 142.55.0.0/16 and it has three campuses situated in Oakville, Brampton and Mississauga in Ontario, Canada. The next schematic summarizes this imaginary topology in the form of routing clouds. The purposes of the BGP sessions will be explained as the configuration is developed, step by step.

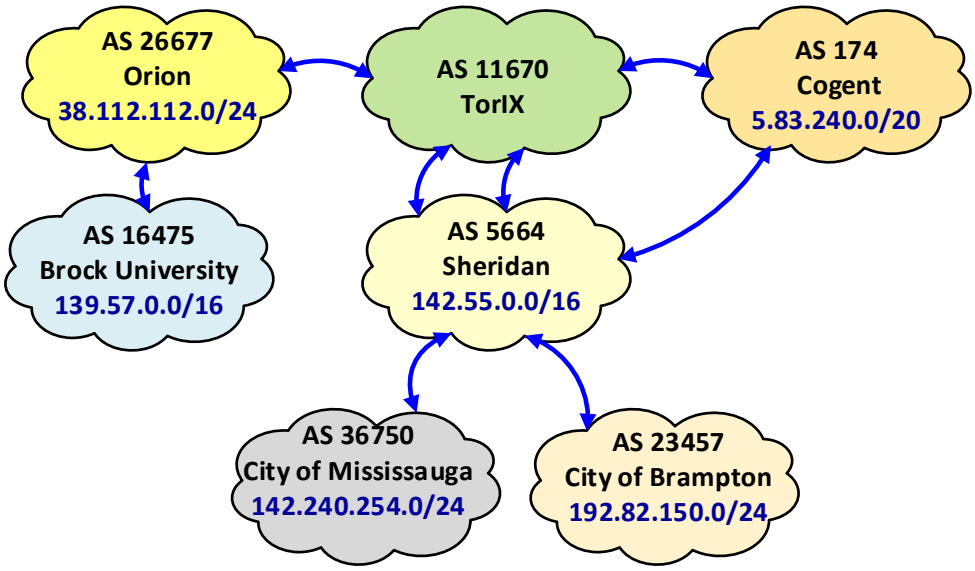


Figure 10.2

In this simulated internetwork, Sheridan gets Internet access via TorIX (AS 11670) and Cogent (AS 174). These two Internet organizations have thousands of other peers which for obvious feasibility reasons cannot be included here.

As per agreement, Sheridan would provide transit for certain networks that belong to the cities of Mississauga (AS 36750) and Brampton (AS 23457). Each of this cities also have their own connections to Internet providers. These additional links will not be deployed in this example topology and thus they are not shown in the diagram.

Additionally, Sheridan connects with Orion (AS 26677) to collaborate with other academic organizations such as Brock University (AS 16475) located in St Catharines, Ontario. Other connection details are omitted as well.

Configuration

AS 5664 Sheridan College

Notice: the configuration is going to concentrate only on BGP assuming that the "skeleton" of the network has already been configured. Hence, the configuration of the interfaces and the IGP routing protocol will be placed in the appendix of this chapter for the readers wanting to practice this case.

The configuration starts by defining the internal BGP sessions among the routers in Sheridan College. Routers R6, R7, and R9 will be autonomous system border routers (ASBR) located in the Mississauga campus. R8 will be the ASBR in the Oakville campus and R10 will be located in the Brampton campus. Leased dark optical fibres would interconnect the three different locations. Internally, the routing protocol OSPF will be configured for IP connectivity while iBGP will be deployed to propagate the external routing knowledge throughout the domain.

A peer group, called INTERNAL, is defined to control BGP inside AS 5664. All the routers peers using their loopback addresses in a full mesh logical topology. Peering to the loopbacks guarantees that there will likely be a path to the neighbour unless that several links fail at the same time. This diagram portrays the logical view of the iBGP peering in AS 5664.

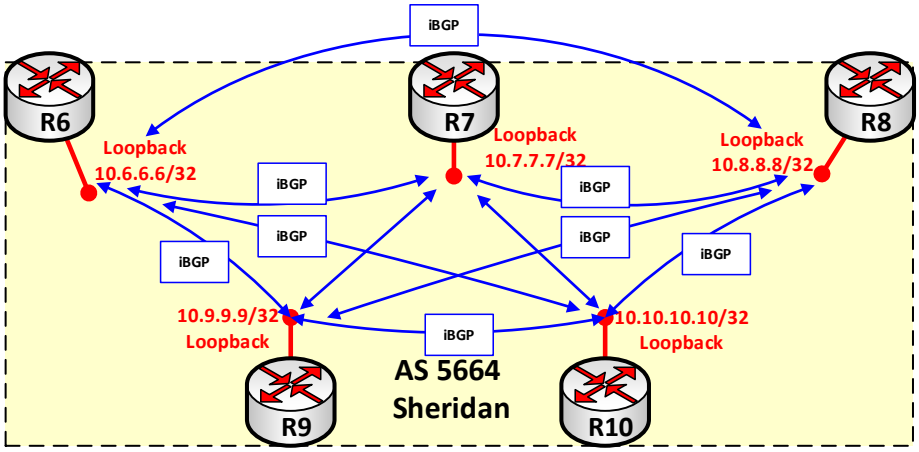


Figure 10.3

A policy to control iBGP is configured as well. Such policy blocks the advertisement of the internal routes via iBGP by matching the regular expression with the empty path string ^$. The internal routes, from the subnetting of the address space 142.55.0.0/16, are to be advertised by the IGP OSPF. Furthermore, the policy allows any other accepted BGP route advertisement coming from the external peers but first, these routes must pass thru the ASBR filters.

This INTERNAL peer-group configuration yields a template to be used in all the routers belonging to AS 5664 with the editing of the loopback addresses, of course. This is such template as applied to router R6.

```
router bgp 5664
neighbor INTERNAL peer-group
neighbor INTERNAL remote-as 5664
neighbor INTERNAL update-source Loopback0
neighbor INTERNAL next-hop-self
neighbor INTERNAL route-map SHERIDAN-iBGP out
neighbor 10.7.7.7 peer-group INTERNAL
neighbor 10.8.8.8 peer-group INTERNAL
neighbor 10.9.9.9 peer-group INTERNAL
neighbor 10.10.10.10 peer-group INTERNAL
```

```
route-map SHERIDAN-iBGP deny 10
match as-path 56
route-map SHERIDAN-iBGP permit 20
```

```
ip as-path access-list 56 permit ^$
```

Figure 10.4

After the internal BGP has been configured, the next step is to start peering with external autonomous systems one by one, starting with the Cities of Mississauga and Brampton.

R9 in AS 5664

Router R9 must accept the routes advertised by the municipal government of the City of Mississauga. This obtained knowledge is to be propagated to the other border routers within Sheridan. These are the requirements for R9:

- Router R9 should not advertise a full Internet routing table to the City Mississauga (AS 36750). Instead, a default route will be enough.
- R9 should only accept routes which are sourced from AS 36750. This is to prevent potential paths coming from the other side since this AS has other connections (not shown in the diagram).
- R9 will propagate the learned routes from Mississauga to routers R6, R7, R8, and R10.

- R9 must not accept martian routes. This is a set of prefixes that should not be advertised, but sometimes, they are advertised because of configuration mistakes.

Here is the configuration of R9 with respect the neighbour R11. There are two polices, one to filter the routing information to be sent to the City of Mississauga and another to filter the information received from this neighbour.

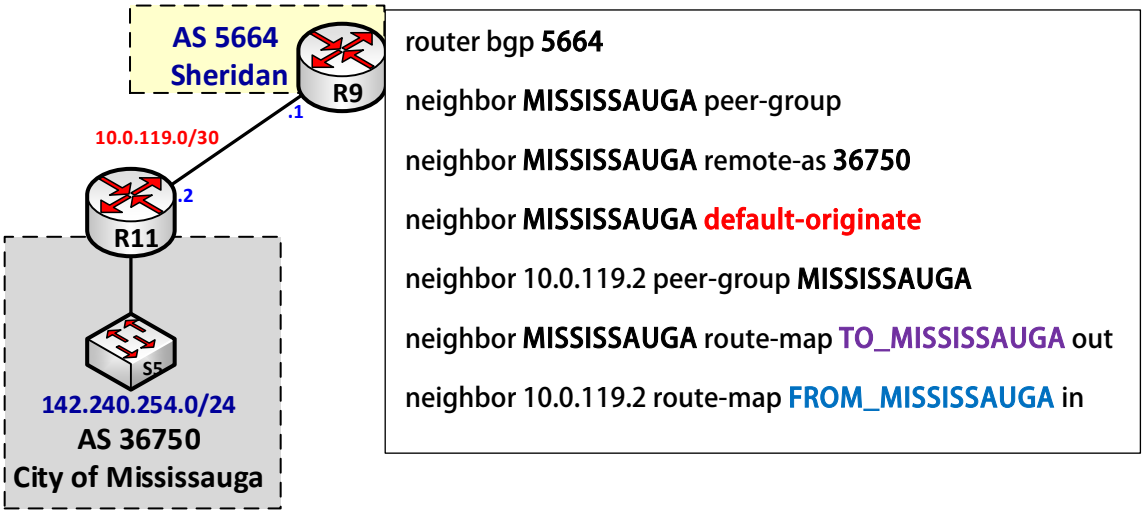


Figure 10.5

The policy TO_MISSISSAUGA fulfills the requirement that Sheridan only advertises a default route to AS 36750. It does so by matching the default prefix only while blocking anything else.

```
route-map TO_MISSISSAUGA permit 10
match ip address prefix-list IPv4_Default_Route_Only
route-map TO_MISSISSAUGA deny 20
```

```
ip prefix-list IPv4_Default_Route_Only seq 5 permit 0.0.0.0/0
```

The policy FROM_MISSISSAUGA conditionally accepts the routes sourced from AS 36750. The City of Mississauga might have external connections to other AS (not shown in diagram) and there is the potential that they could see a transit path via Mississauga, then Sheridan to some destinations. This depends on what each administration does and that is beyond the local control. Consequently, this policy will avoid that possibility by matching only the routes originated in AS 36750. Another version of this policy could exactly match the IP prefixes instead of the AS number, but that might require periodic updating to add or remove prefixes.

There is another potential problem that must be prevented. Matching only the AS number is a loose policy because the administration in the neighbour AS could send, by mistake, the wrong routing information such as private IP address space, for example.

There is a standard filter, well-known in the Internet community, which solves this problem. NO-MARTIANS blocks default routes, all private IP address space, the testing prefix 192.0.2.0/24, the automatic private IP address 169.254.0.0/16, the multicast IP address space and the Class E address space. Whatever else that survives the filter is allowed.

```
ip prefix-list NO-MARTIAN seq 5 deny 0.0.0.0/8 le 32
ip prefix-list NO-MARTIAN seq 10 deny 0.0.0.0/0
ip prefix-list NO-MARTIAN seq 15 deny 10.0.0.0/8 le 32
ip prefix-list NO-MARTIAN seq 20 deny 127.0.0.0/8 le 32
ip prefix-list NO-MARTIAN seq 25 deny 169.254.0.0/16 le 32
ip prefix-list NO-MARTIAN seq 30 deny 172.16.0.0/12 le 32
ip prefix-list NO-MARTIAN seq 35 deny 192.0.2.0/24 le 32
ip prefix-list NO-MARTIAN seq 40 deny 192.168.0.0/16 le 32
ip prefix-list NO-MARTIAN seq 45 deny 224.0.0.0/4 le 32
ip prefix-list NO-MARTIAN seq 50 deny 240.0.0.0/4 le 32
ip prefix-list NO-MARTIAN seq 55 permit 0.0.0.0/0 le 32
```

This is the policy to be applied to the routing information coming from Mississauga.

```
route-map FROM_MISSISSAUGA permit 10
 match ip address prefix-list NO-MARTIAN
 match as-path 36
!
route-map FROM_MISSISSAUGA deny 20
!
```

```
ip as-path access-list 36 permit ^36750$
```

A more complete policy would also include blocking the Bogons. These are address spaces that have not been allocated yet to any organization so they should not appear in the Internet routing tables. However, they show up occasionally because of configuration mistakes or when addresses are used in a malicious way to source an attack, for example. Bogons have been disappearing as the IPv4 address space have been largely allocated. When a Bogon space is assigned to an organization, it ceases to be Bogon. For that reason, routing policies that block Bogons have to be updated [6] frequently.

R11 in AS 36750

The configuration in the side of the City of Mississauga is kept very simple.

```
router bgp 36750
network 142.240.254.0 mask 255.255.255.0
neighbor 10.0.119.1 remote-as 5664
```

At this point, the BGP tables can be verified in both ends. First, router R9 has learned the prefix 142.240.254.0/24 advertised by AS

R9-Sheridan#show ip bgp (edited)

Network	Next Hop	Metric	LocalPref	Path	Organization
0.0.0.0	0.0.0.0			i	Sheridan (internal)
*> 142.240.254.0/24	10.0.119.2	0		**36750** i	**City of Mississauga**

On the other side, router R11 has learned a default route being advertised by Sheridan.

R11-Mississauga#show ip bgp (edited)

Network	Next Hop	Metric	LocalPref	Path	Organization
*> 0.0.0.0	10.0.119.1			**5664** i	**Sheridan**
*> 142.240.254.0/24	0.0.0.0	0		i	City of Mississauga (local)

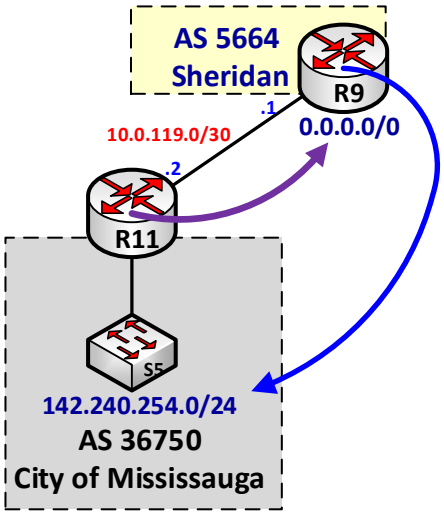


Figure 10.6

R10 in AS 5664

This router faces the autonomous system 23457 which represents the municipal government of the City of Brampton. Essentially, it has the same configuration profile than R9, the only differences being the specific addresses and the AS number.

Sheridan will feed a default route with the policy TO_BRAMPTON and it will accept the routes sourced from AS 23457, while preventing the propagation of Martian routes with the policy FROM_BRAMPTON.

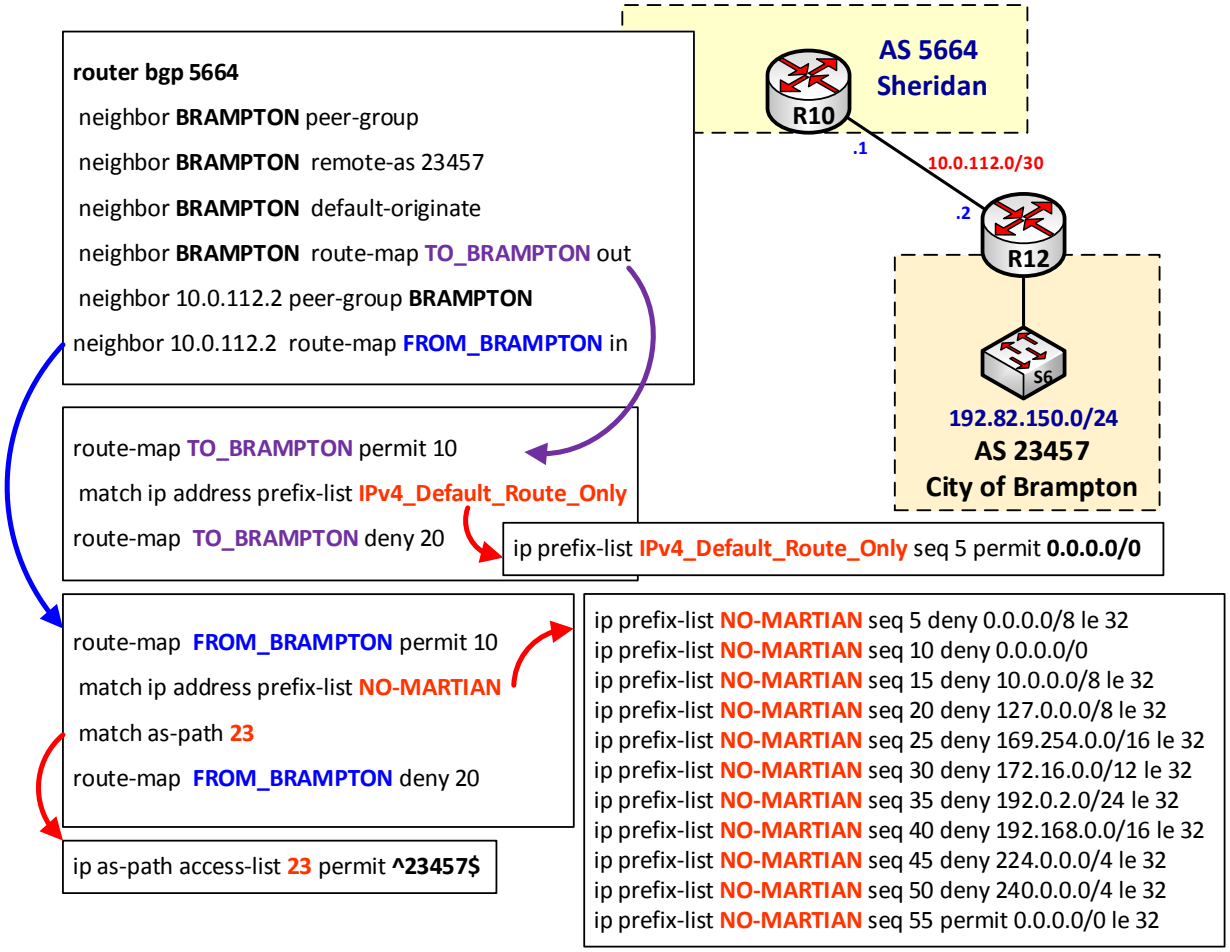


Figure 10.7

R12 in AS 23475

The City of Brampton router is left with a very basic configuration.

```
router bgp 23457
network 192.82.150.0 mask 255.255.255.0
neighbor 10.0.112.1 remote-as 5664
```

As a result, router R10 learns the prefix 192.82.150.0/24 advertised by Brampton. Also, since this router iBGP peers with R9, it has learned the Mississauga route 142.240.254.0/24.

R10-Sheridan#show ip bgp (edited)

Network	Next Hop	Metric	LocalPref	Path	Organization
0.0.0.0	0.0.0.0			i	Sheridan (internal)
*>i 142.240.254.0/24	10.9.9.9	0	100	36750 i	City of Mississauga
*> 192.82.150.0	10.0.112.2	0		23457 i	City of Brampton

On the other side, Brampton has learned a default route only from Sheridan.

R12-Brampton#show ip bgp (edited)

Network	Next Hop	Metric	LocalPref	Path	Organization
*> 0.0.0.0	10.0.112.1			5664 i	Sheridan
*> 192.82.150.0/24	0.0.0.0	0		i	City of Brampton (local)

Since this part of the topology have been successfully completed, it is time to move on to another part, starting with Sheridan's AS border router R6.

R6 in AS 5664

ASBR R7 has three peering profiles. One is the INTERNAL (iBGP) template already created and explained. Another profile is for the peering with TorIX and one more for Orion.

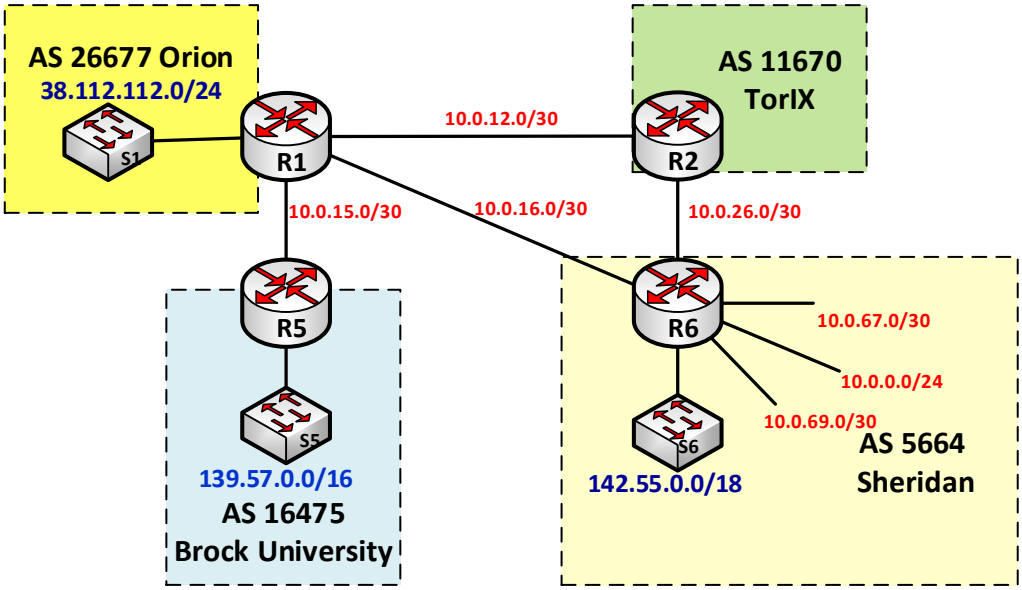


Figure 10.8

These are the requirements for peering with Orion.

For Advertising to Orion:

- R6 must advertise the aggregation 142.55.0.0/16 for the address space of Sheridan to R1 in Orion.
- R6 will advertise the routes from the cities of Mississauga and Brampton to Orion.
- R6 should mark the advertised routes with the community string needed by Orion.

For Receiving from Orion:

- R6 should accept the routes from Orion if they are marked with the correct communities.
- Then R6 should propagate such routes throughout AS 5664 using iBGP.
- R6 will be the preferred exit point for these routes in case that they were learned from another via. To accomplish that, R6 will mark the routes with a high LOCAL_PREFERENCE value of 606. (Six is just used as a mnemonic value for R6)

R6 Configuration

Router R6 creates an aggregate route 142.55.0.0/16 but in Cisco IOS, an aggregate is only advertised if there is at least one subnet prefix that activates it. For that reason, the network command includes the prefixes 142.55.0.0/18 and 142.55.64.0/18. If any of these two networks is present in the routing table, then the aggregate will be advertised.

```
router bgp 5664
 network 142.55.0.0 mask 255.255.192.0
 network 142.55.64.0 mask 255.255.192.0
 aggregate-address 142.55.0.0 255.255.0.0
```

A peer-group ORION is created for this peering profile. Two policies are added, one to advertise (TO_ORION) and another to receive (FROM_ORION).

```
neighbor ORION remote-as 26677
neighbor ORION send-community
neighbor ORION route-map TO_ORION out
neighbor 10.0.16.1 peer-group ORION
neighbor 10.0.16.1 route-map FROM_ORION in
```

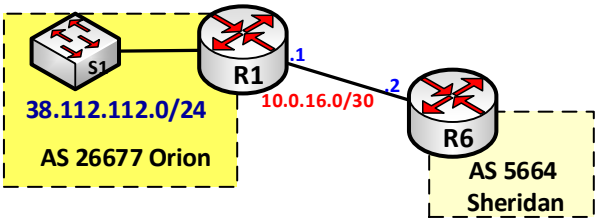


Figure 10.9

This is the policy TO_ORION.

```
route-map TO_ORION permit 10
 description "advertise 142.55/16 to Orion"
 match ip address prefix-list AGGREGATE
 set community 26677:10
!
route-map TO_ORION permit 20
 description "Brampton routes to Orion"
 match as-path 23
 set community 26677:10
!
route-map TO_ORION permit 30
 description "Mississauga routes to Orion"
 match as-path 36
 set community 26677:10
!
route-map TO_ORION deny 40
 description "deny anything else"
```

```
ip prefix-list AGGREGATE seq 5 permit 142.55.0.0/16
```

```
ip as-path access-list 23 permit _23457$
```

```
ip as-path access-list 36 permit _36750$
```

The prefix list AGGREGATE matches uniquely 142.55.0.0/16 and nothing else. The other statements in the route-map match the prefixes originated by Brampton and Mississauga. Notice that the community 26677:10 is being attached to all the prefixes being advertised and there is a reason for that.

Orion specifies in a public document [5] available in its website [4] that the community 26677:10 is assigned to its member's routes. In fact, Sheridan College and Brock University are members of this community. By marking the routes with this tag, the policies are simplified everywhere because it is a matter of just matching the string to either accept or reject routes.

Regarding the inbound updates, R9 will accept the routes advertised by Orion, only if they come with the communities specified by Orion. This is done with the very simple policy FROM_ORION. Two strings are to be matched, the member's routes [5] community 26677:10 and community 26677:20 which is set by Orion to its own internal routes (like its data centre routes, for example). Next, this is the policy in R6.

```
!
route-map FROM_ORION permit 10
 match community ORION_COMM
 set local-preference 606
!
```

```
ip community-list expanded ORION_COMM permit 26677:10
ip community-list expanded ORION_COMM permit 26677:20
```

As an alternative, this policy achieves the same result but in a more granular way.

```
ip community-list expanded ORION-MEMBERS permit 26677:10
ip community-list expanded ORION-ROUTE permit 26677:20
!
route-map FROM_ORION permit 10
 match community ORION-MEMBERS
 set local-preference 606
!
route-map FROM_ORION permit 20
 match community ORION-ROUTE
 set local-preference 606
!
```

This peering profile is complete on the side of R6 at Sheridan. Now, the configuration continues on the side of Orion's router R1.

Orion AS 26677

The Ontario Research and Innovation Optical Network [4] interconnects educational, research, and government institutions via a large network of optical fibre. Although Orion peers with Internet providers, its main function is not to provide commercial Internet access but mostly to facilitate collaboration among its members. That is exactly the way that Sheridan and Brock University will interact in this pretend-network. They will mutually advertise some prefixes to establish a community of traffic interest via Orion.

R1 in AS 26677

This router has two main configuration profiles even though there are three peering organizations. Both Sheridan and Brock have the same profile because they have the same characteristics as members of the Orion community. The relationship with TorIX is different though because that is a transit service to other autonomous systems.

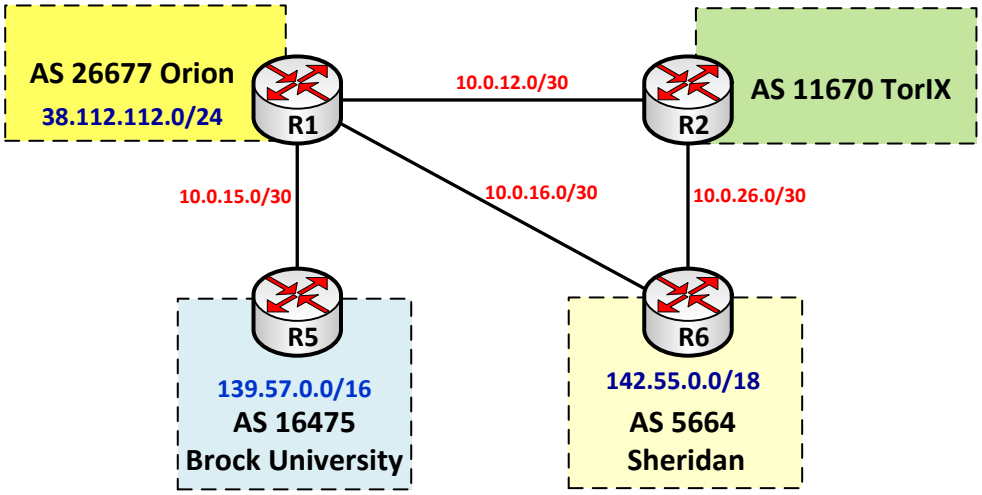


Figure 10.10

As mentioned before, Orion has a well-defined structure of community strings which is publicly [5] available in its website. The use of the communities greatly simplifies routing policies because it is a matter of matching the strings instead of matching individual prefixes. Orion assigns 26677:10 to member's routes and 26677:20 to its own routes.

Orion applies an inbound policy profile to Sheridan to accept the routes sourced in AS 5664 and then it marks the routes with the community 26677:10. (it is remarking just in case that the member were not marking the routes. In this configuration, Sheridan's R6 has already set its community string.)

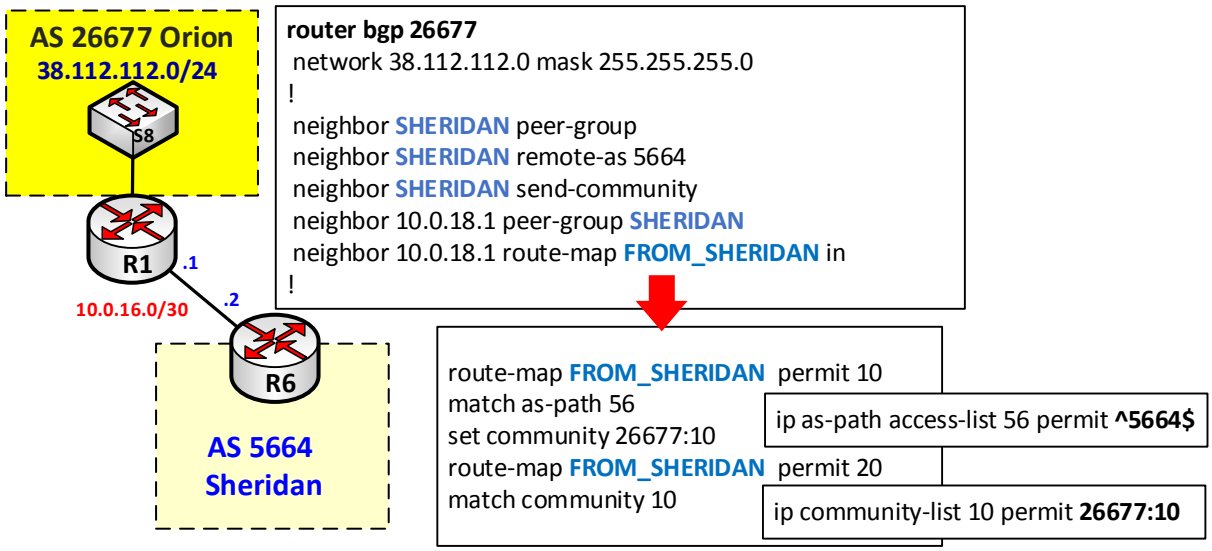


Figure 10.11

Orion sets an outbound policy to send its own routes and members routes to Sheridan. These routes have the corresponding communities 26677:20 and 26677:10 as shown next.

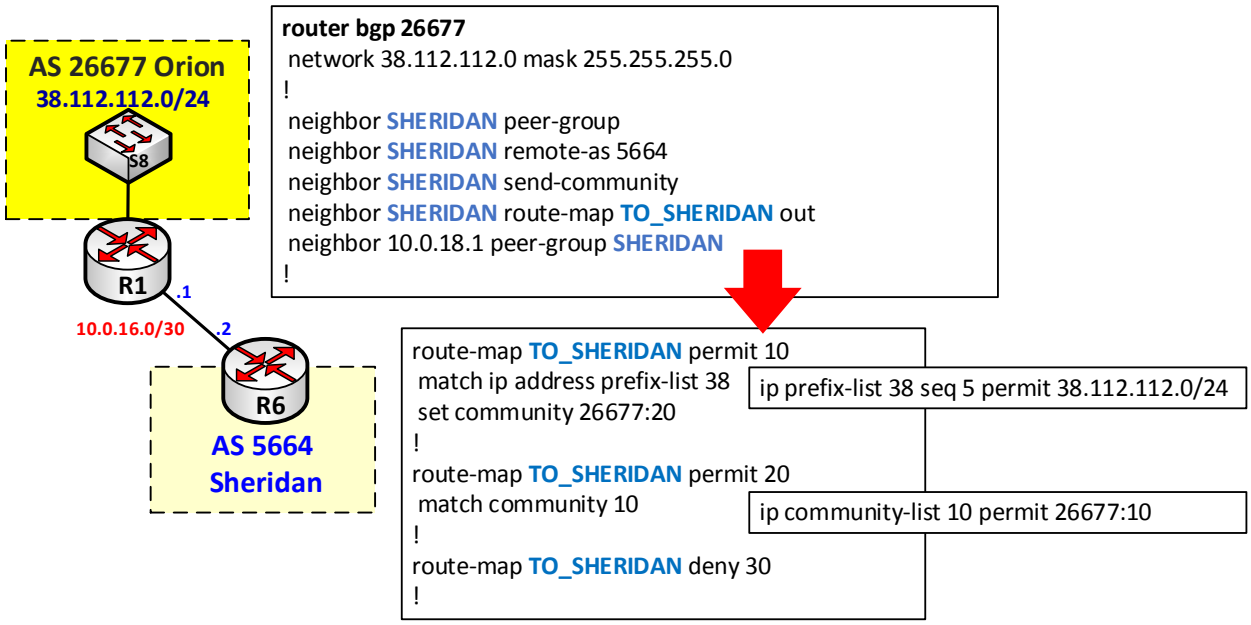


Figure 10.12

In the pretend network, Orion applies this inbound policy toward Brock University. Such policy is very tight and it accepts only the routes sourced from Brock by matching the AS number AS 16475 exactly.

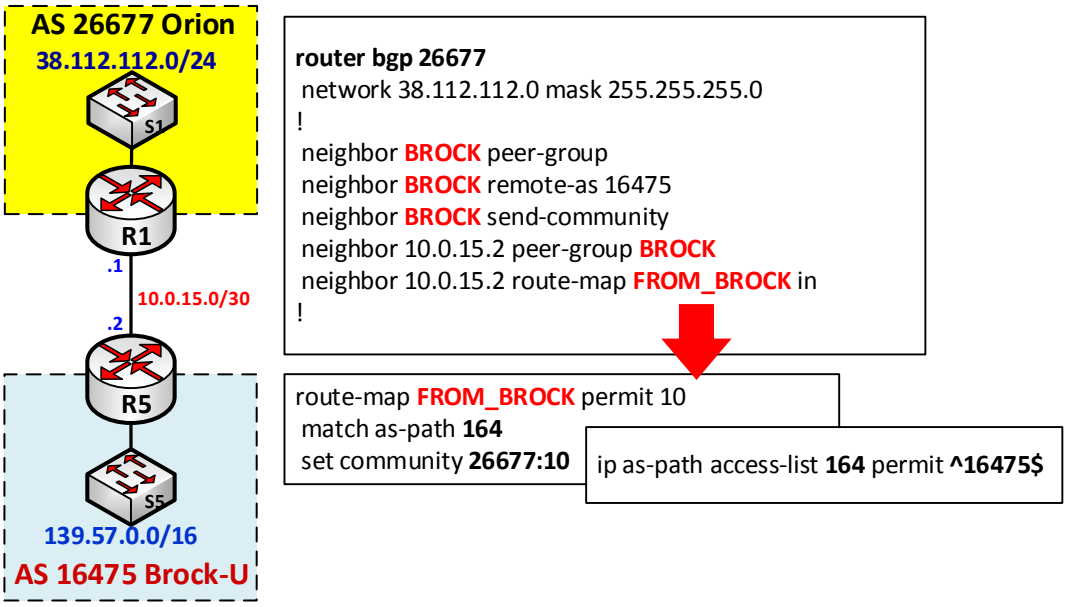


Figure 10.13

Orion also applies an outbound policy that will only advertise the route from Orion and also from members of the Orion community.

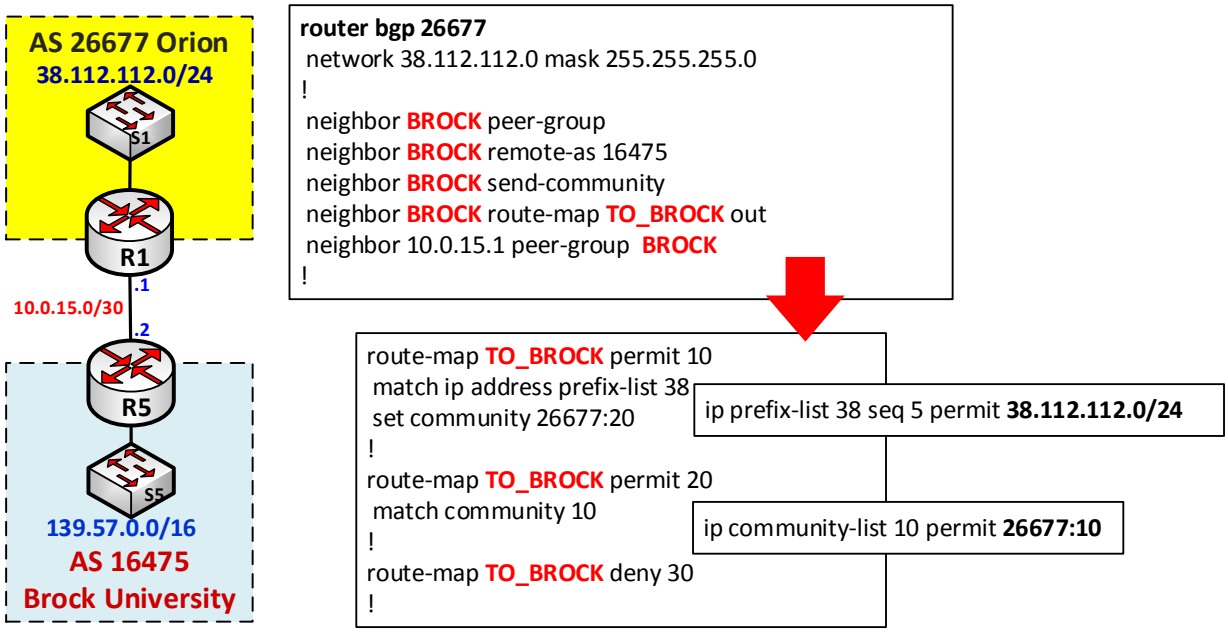


Figure 10.14

The policy TO_BROCK matches the prefix 38.112.112.0/24 and it sets the community 26677:20 indicating that it is a route that belongs to Orion. The second statement allows routes that have been marked with the community 26677:10 to be advertised to Brock.

It is necessary to indicate that the previous policy FROM_BROCK has set the community 26677:10 to 139.57.0.0/16, but R1 knows that this route is coming from AS16475 and it will not advertise it back (split horizon rule). However, routes that are coming from other members of the community, such as Sheridan College, will be advertised to Brock. This functioning will be verified once that all the policies are in place.

Router R5, AS 16475, Brock University

In real life, this University has a more complex interconnection to Internet, but in this made-up network, it has been simplified to make the explanation feasible. Thus, Brock just advertises the address space 139.57.0.0/16 to Orion without even marking with the community 26677:10. But it does not matter because Orion already has the policy to add the community string.

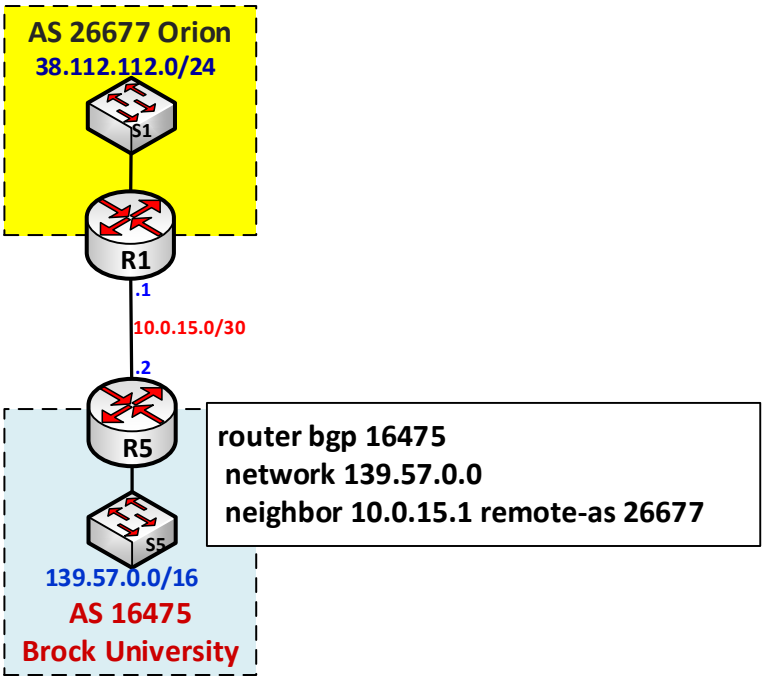


Figure 10.15

Verification

The functioning of the policies implemented so far can be confirmed. Router R1 knows the following:

R1-Orion#show ip bgp (edited)

Network	Next Hop	Metric	AS_Path	Organizations in AS_Path
*> 38.112.112.0/24	0.0.0.0	0	i	Orion (internal)
*> 139.57.0.0	10.0.15.2	0	16475 i	Brock University
*> 142.55.0.0	10.0.16.2	0	5664 i	Sheridan
*> 142.240.254.0/24	10.0.16.2		5664 36750 i	Sheridan City of Mississauga
*> 192.82.150.0	10.0.16.2		5664 23457 i	Sheridan City of Brampton

Orion's router has learned the route advertised by Brock University, the aggregation from Sheridan and the routes from the cities that passed thru Sheridan. This knowledge is all correct and it can be better visualized in this diagram:

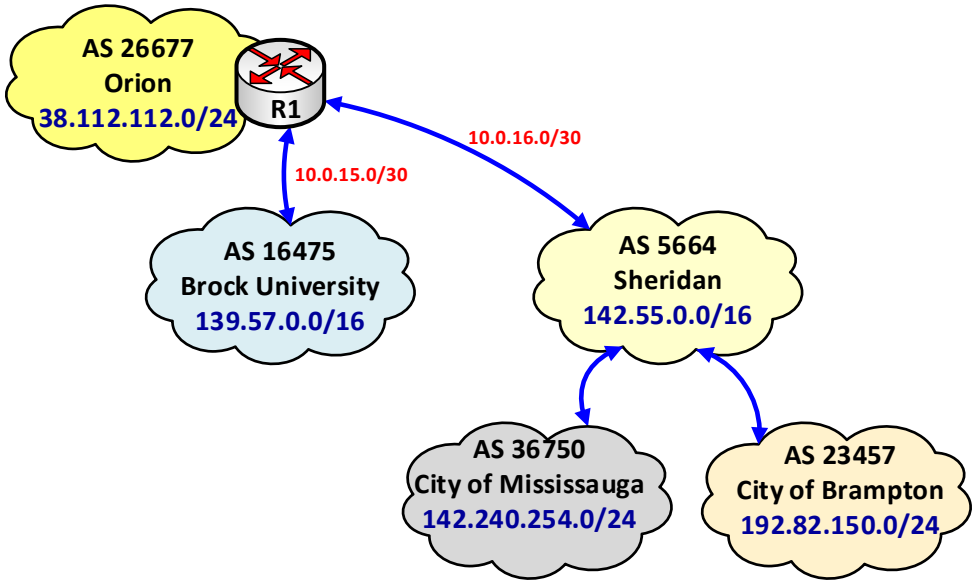


Figure 10.16

This proves that not only the configuration of Orion and Brock are correct, but also that the Sheridan's settings are correct; otherwise, the routes from Mississauga and Brampton would not have arrived to Orion.

In the meantime, R5 in Brock University has learned the following:

```
R5-Brock#show ip bgp (edited)
   Network          Next Hop     Metric  Path                 Organizations in AS_Path
*> 38.112.112.0/24  10.0.15.1     0      26677 i              Orion
*> 139.57.0.0       0.0.0.0       0      i                    Brock University (internal)
*> 142.55.0.0       10.0.15.1            26677 5664 i         Orion Sheridan
*> 142.240.254.0/24 10.0.15.1            26677 5664 36750 i   Orion Sheridan Mississauga
*> 192.82.150.0     10.0.15.1            26677 5664 23457 i   Orion Sheridan Brampton
```

Furthermore, the aggregation prefix from Sheridan contains the community string assigned to Orion members as this command can prove it:

```
R5-Brock#show bgp ipv4 unicast 142.55.0.0/16
Paths: (1 available, best #1, table default) Not advertised to any peer
26677 5664, (aggregated by 5664 10.1.1.1) coming from 10.0.15.1 (10.1.1.1)
Origin IGP, localpref 100, valid, external, atomic-aggregate, best
Community: 26677:10
```

The prefix sourced at Orion contains the community string for Orion routes:

```
R5-Brock#show bgp ipv4 unicast 38.112.112.0/24
Paths: (1 available, best #1, table default) Not advertised to any peer
26677 coming from 10.0.15.1 (10.1.1.1)
Origin IGP, metric 0, localpref 100, valid, external, best
Community: 26677:20
```

The routes from the cities also are marked with the Orion member community:

```
R5-Brock#show bgp ipv4 unicast 142.240.254.0/24
26677 5664, coming from 10.0.15.1 (10.1.1.1)
Community: 26677:10
```

```
R5-Brock#show bgp ipv4 unicast 192.82.150.0/24
26677 5664, (coming from 10.0.15.1 (10.1.1.1)
Community: 26677:10
```

Brock receives from Orion, the routes marked with the communities and nothing else. Which is correct according to the intention of the policy.

This is the result of the policies applied by Orion toward Sheridan as observed in R6.

```
R6#show bgp ipv4 unicast 38.112.112.0/24 (edited)
AS_PATH 26677, from 10.0.16.1 (10.1.1.1). Origin IGP, metric 0, localpref 606,
valid, external, best.
Community: 26677:20

R6#show bgp ipv4 unicast 139.57.0.0/16 (edited)
AS_PATH 26677 16475, from 10.0.16.1 (10.1.1.1). Origin IGP, localpref 606,
valid, external, best
Community: 26677:10
```

The route sourced in Orion has the correct community 26677:20 as it was set up by Orion while the route from Brock has the member community 26677:10. Both routes have the Local_Preference value set to 606 as per the policy in router R6.

Configuration Orion and TorIX

Orion advertises to TorIX only its internal routes. A simple, loose policy matches the empty path only. In this way, Orion prevents being a potential transit path between TorIX and Sheridan, for example, in case that the link between TorIX and Sheridan would fail. (The policy is intended to be like this although, in theory, it could be that Orion would offer a transit path, but that is not the function of Orion anyway.) This completes the configuration of R1.

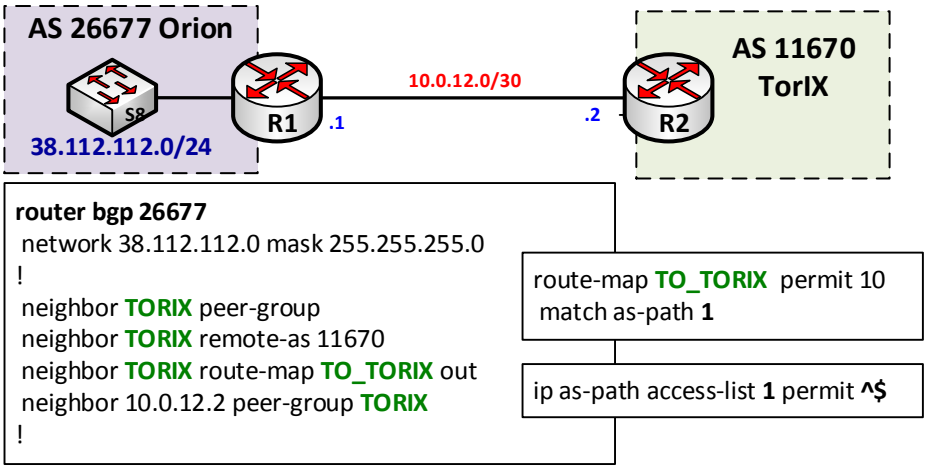


Figure 10.17

TorIX AS 11670

Torix is both an autonomous system and also the Toronto Internet Exchange Point which is distributed into two buildings [3] (151 Front St. and 45 Parliament St. in Toronto) with the infrastructure to act as a carrier hotel [3] so organizations can interconnect. In this make-believe topology, Sheridan peers with Torix which is providing transit to many other AS (not shown).

Router R2 in AS 11670

Since this case study is concentrated on the enterprise side (Sheridan) the TorIX routers will have simple configurations just to allow interconnections. R2 has three neighbour profiles which includes iBGP to talk to router R3. The other profiles are facing Orion and Sheridan. There is only one loose policy which just blocks Martian routes. (In real life, a router like this would have a very strict and complicated policy).

```
router bgp 11670
 neighbor SHERIDAN peer-group
 neighbor SHERIDAN remote-as 5664
 neighbor INTERNAL peer-group
 neighbor INTERNAL remote-as 11670
 neighbor INTERNAL next-hop-self
 neighbor ORION peer-group
 neighbor ORION remote-as 26677
 neighbor 10.0.12.1 peer-group ORION
 neighbor 10.0.23.2 peer-group INTERNAL
 neighbor 10.0.26.2 peer-group SHERIDAN
 neighbor 10.0.26.2 route-map CLIENT in
!
route-map CLIENT permit 10
 match ip address prefix-list NO-MARTIAN
!
```

```
ip prefix-list NO-MARTIAN seq 5 deny 0.0.0.0/8 le 32
ip prefix-list NO-MARTIAN seq 10 deny 0.0.0.0/0
ip prefix-list NO-MARTIAN seq 15 deny 10.0.0.0/8 le 32
ip prefix-list NO-MARTIAN seq 20 deny 127.0.0.0/8 le 32
ip prefix-list NO-MARTIAN seq 25 deny 169.254.0.0/16 le 32
ip prefix-list NO-MARTIAN seq 30 deny 172.16.0.0/12 le 32
ip prefix-list NO-MARTIAN seq 35 deny 192.0.2.0/24 le 32
ip prefix-list NO-MARTIAN seq 40 deny 192.168.0.0/16 le 32
ip prefix-list NO-MARTIAN seq 45 deny 224.0.0.0/4 le 32
ip prefix-list NO-MARTIAN seq 50 deny 240.0.0.0/4 le 32
ip prefix-list NO-MARTIAN seq 55 permit 0.0.0.0/0 le 32
```

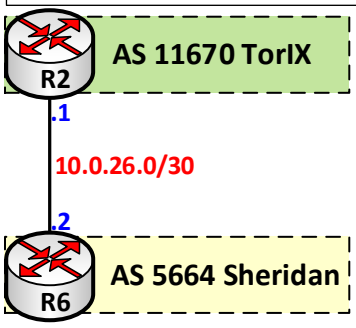


Figure 10.18

Router R3 in AS 11670

Router R3 has also an internal profile for iBGP, another peer-group for Sheridan, and another to peer with Cogent. Again, the routing policy has been simplified to a minimum.

```
router bgp 11670
neighbor SHERIDAN peer-group
neighbor SHERIDAN remote-as 5664
neighbor INTERNAL peer-group
neighbor INTERNAL remote-as 11670
neighbor INTERNAL next-hop-self
neighbor COGENT peer-group
neighbor COGENT remote-as 174
neighbor 10.0.23.1 peer-group INTERNAL
neighbor 10.0.34.2 peer-group COGENT
neighbor 10.0.37.2 peer-group SHERIDAN
neighbor 10.0.37.2 route-map CLIENT in
!
route-map CLIENT permit 10
match ip address prefix-list NO-MARTIAN
!
```

```
ip prefix-list NO-MARTIAN seq 5 deny 0.0.0.0/8 le 32
ip prefix-list NO-MARTIAN seq 10 deny 0.0.0.0/0
ip prefix-list NO-MARTIAN seq 15 deny 10.0.0.0/8 le 32
ip prefix-list NO-MARTIAN seq 20 deny 127.0.0.0/8 le 32
ip prefix-list NO-MARTIAN seq 25 deny 169.254.0.0/16 le 32
ip prefix-list NO-MARTIAN seq 30 deny 172.16.0.0/12 le 32
ip prefix-list NO-MARTIAN seq 35 deny 192.0.2.0/24 le 32
ip prefix-list NO-MARTIAN seq 40 deny 192.168.0.0/16 le 32
ip prefix-list NO-MARTIAN seq 45 deny 224.0.0.0/4 le 32
ip prefix-list NO-MARTIAN seq 50 deny 240.0.0.0/4 le 32
ip prefix-list NO-MARTIAN seq 55 permit 0.0.0.0/0 le 32
```

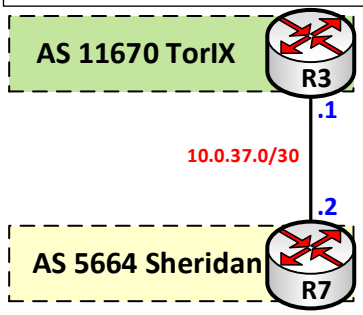


Figure 10.19

Since TorIX is completed, then the configuration goes back to R6 with the corresponding peering profile for TorIX.

```
neighbor TORIX peer-group
neighbor TORIX remote-as 11670
neighbor 10.0.26.1 peer-group TORIX neighbor
TORIX route-map TO_TORIX out
neighbor 10.0.26.1 route-map FROM_TORIX in
```

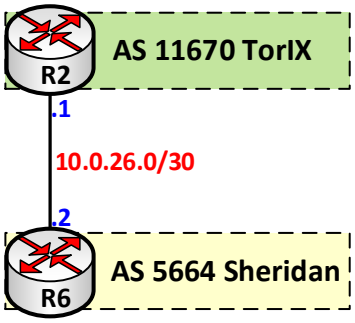


Figure 10.20

Router R6 AS 5664 Continuation

R6 will have a full Internet routing table so it accepts whatever TorIX advertises and it sets a local-preference value of 66 to the routes obtained via TorIX's routers R2. The objective of setting the local preference is to manipulate the exit point toward Orion from the viewpoint of the routers in Sheridan.

```
route-map FROM_TORIX permit 10
 set local-preference 66
```

When R6 further propagates these routes to the other routers inside AS 5664, it will do so with a local preference of 66 as shown in the diagram.

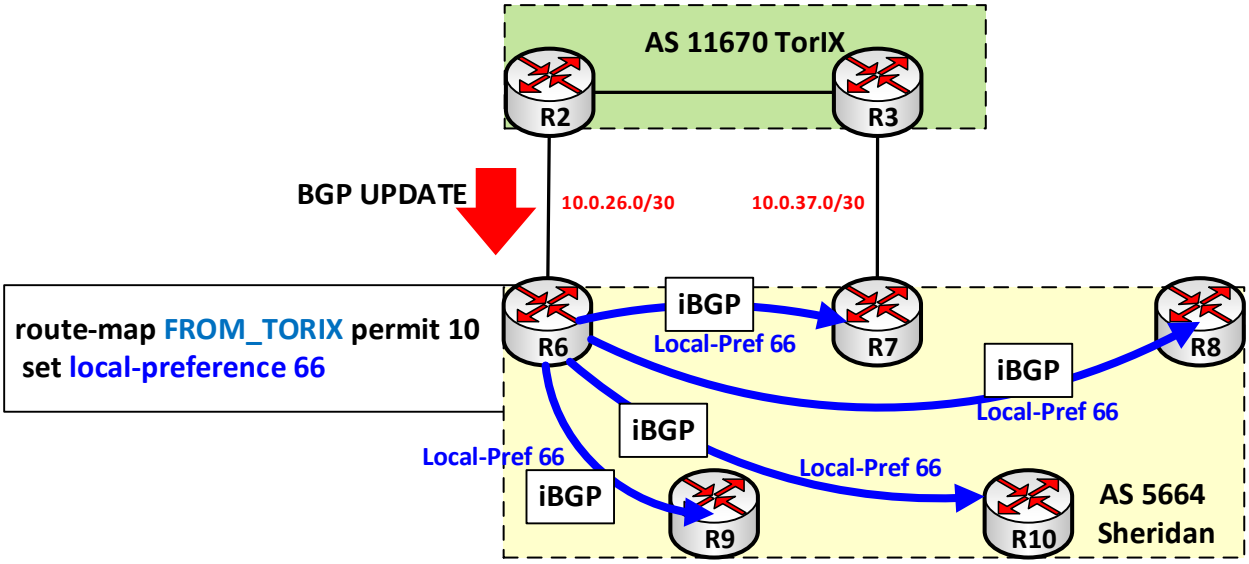


Figure 10.21

Sheridan has two direct paths to TorIX and one of them should be chosen as the preferred exit point. The idea is that the other routers inside Sheridan will compare R6's local-preference value to the one advertised by R7 (a higher value when this router gets configured later). The highest local preference wins. In such case, R7 will be the preferred path.

Regarding the outbound updates, R6 advertises to TorIX the aggregate 142.55.0.0/16 with the MED attribute set to 10. Later, R7 will get configured with no particular action for the metric. Since the aggregate is created in the router, there is no metric associated to it by default. Consequently, traffic destined to 142.55.0.0/16 flowing from TorIX should take the link 10.0.37.0/30 that offers the lowest metric.

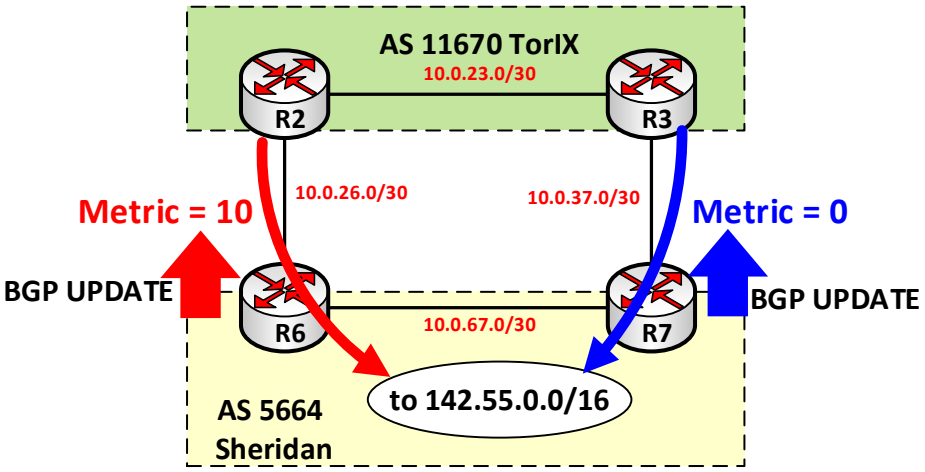


Figure 10.22

This is the policy statement to implement the routing intention.

```
route-map TO_TORIX permit 10
 description "advertise 142.55/16 to Torix HI-MED"
 match ip address prefix-list AGGREGATE
 set metric 10
!
route-map TO_TORIX permit 20
 description "Brampton routes to Torix HI-MED"
 match as-path 23
 set metric 10
!
route-map TO_TORIX permit 30
 description "Mississauga routes to Torix LO-MED"
 match as-path 36
!
route-map TO_TORIX deny 40
 description "deny anything else
!
```

```
ip prefix-list AGGREGATE seq 5 permit 142.55.0.0/16
```

```
ip as-path access-list 23 permit _23457$
```

```
ip as-path access-list 36 permit _36750$
```

Additionally, the policy matches the routes sourced from Brampton and its sets a metric of 10. When router R7 is configured the metric for the same route would be left to a default value which is lower than 10. In this way, the neighbour router R2 would find more appealing the internal path to R3 to get to the network of Brampton.

On the other hand, Mississauga's route is left without a specific metric while R7 will advertise a higher metric for the same route. So the path via R6 will be more attractive to TorIX than the path via R7. The final statement closes the policy and nothing else is advertised by router R6.

It is necessary to point out that this would only work if there is agreement with TorIX to accept these metrics because the administration of AS 11670 can easily place their own policy to erase or change the metrics. In fact, many ISPs have a default policy to erase metrics. Basically, the customer does not dictate policy to the provider.

This completes the configuration of router R6. To actually observe the results it is necessary to continue with the configuration of other routers.

AS 11670 TorIX

Until this point in the make-believe topology, AS16475 (Brock-University), AS 26677 (Orion) and a section of AS5664 have been configured. It is time to move on to another part of the network, specifically the configuration of TorIX.

These will be the policies to be implemented in the TorIX fake network:

- AS 11670 can provide transit between Orion and Cogent.
- AS 11670 can provide transit between Orion and Sheridan.
- AS 11670 can provide transit between Cogent and Sheridan.
- AS 11670 will have a loose inbound traffic policy with respect Sheridan to allow the traffic from the administrations of the City of Mississauga and the City of Brampton. However, TorIX must prevent that bogus routes be propagated from either Sheridan, Mississauga or Brampton.

The next partial view of the diagram represents the traffic interest of TorIX so unnecessary details have been removed.

Routers R9 and R10 in AS11670

Both TorIX routers have similar configuration.

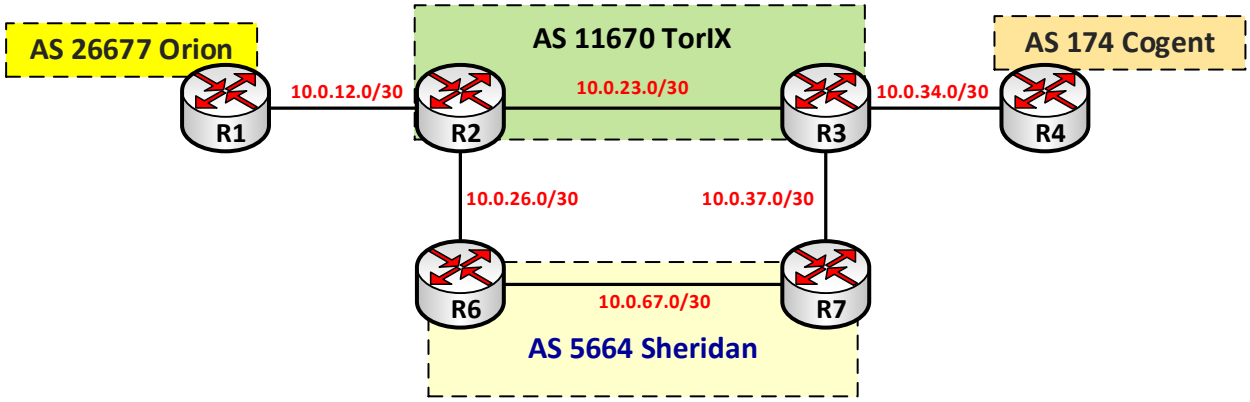


Figure 10.23

```
hostname R2
!
router bgp 11670
neighbor SHERIDAN peer-group
neighbor SHERIDAN remote-as 5664
neighbor  10.0.26.2  peer-group  SHERIDAN
neighbor 10.0.26.2 route-map CLIENT in

neighbor ORION peer-group
neighbor ORION remote-as 26677
neighbor 10.0.12.1 peer-group ORION

neighbor INTERNAL peer-group
neighbor INTERNAL remote-as 11670
neighbor INTERNAL next-hop-self
neighbor 10.0.23.2 peer-group INTERNAL
!
route-map CLIENT permit 10
 match ip address prefix-list NO-MARTIAN
!
route-map CLIENT permit 20
match as-path 56
!
ip as-path access-list 56 permit _5664_
!
```

```
hostname R3
!
router bgp 11670
neighbor SHERIDAN peer-group
neighbor SHERIDAN remote-as 5664
neighbor 10.0.37.2 peer-group SHERIDAN
neighbor 10.0.37.2 route-map CLIENT in

 neighbor COGENT peer-group
 neighbor COGENT remote-as 174
 neighbor 10.0.34.2 peer-group COGENT

neighbor INTERNAL peer-group
neighbor INTERNAL remote-as 11670
neighbor INTERNAL next-hop-self
neighbor 10.0.23.1 peer-group INTERNAL
!
route-map CLIENT permit 10
 match ip address prefix-list NO-MARTIAN
!
route-map CLIENT permit 20
match as-path 56
!
ip as-path access-list 56 permit _5664_
!
```

R7 in AS 5664

This is an interesting router. Located in the Mississauga Campus in the make-believe topology, this device would be considered the preferred gateway to Sheridan's network. In effect, it has a high speed link to TorIX dual and high speed links to Cogent as observed in next partial view.

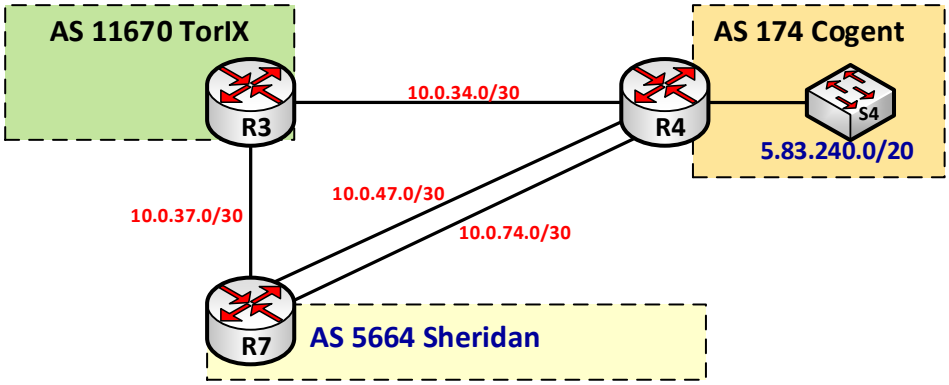


Figure 10.25

These are the routing intentions for R7:

- Sheridan would like that the traffic from TorIX to 142.55.0.0/16 to flow from R3 to R7. This is achieved by advertising a low metric from R7 while R6 is already advertising a higher metric (10) for this route.

- Sheridan would like that the traffic inbound to Mississauga and Brampton to come via TorIX. This is done by advertising the routes to TorIX, but not to Cogent.

- The traffic from TorIX to the City of Mississauga should take the path via R2 then R6. To do so, a lower metric is already advertised by R6 and a higher metric (10) will be advertised by R7.

- The traffic from TorIX to the City of Brampton should take the path via R3 then R7. To do so, R6 is already advertising a higher metric (10) while R7 shows zero metric.

- Sheridan would like to do traffic load balancing with the two links interconnecting to Cogent. This is done by activating multiple BGP paths.

Router R7 in AS 5664

This is the complete implementation of router R7.

```
router bgp 5664
 network 142.55.0.0 mask 255.255.192.0
 network 142.55.64.0 mask 255.255.192.0
 network 142.55.128.0 mask 255.255.192.0
 network 142.55.192.0 mask 255.255.192.0
 aggregate-address 142.55.0.0 255.255.0.0
!
neighbor TORIX peer-group
 neighbor TORIX remote-as 11670
 neighbor TORIX route-map TO_TORIX out
 neighbor 10.0.37.1 peer-group TORIX
!
 neighbor COGENT peer-group
 neighbor COGENT remote-as 174
 neighbor COGENT route-map TO_COGENT out
 neighbor 10.0.47.1 peer-group COGENT
 neighbor 10.0.74.1 peer-group COGENT
!
maximum-paths 3
!
 neighbor INTERNAL peer-group
 neighbor INTERNAL remote-as 5664
 neighbor INTERNAL update-source Loopback0
 neighbor INTERNAL next-hop-self
 neighbor INTERNAL route-map SHERIDAN_iBGP out
 neighbor 10.6.6.6 peer-group INTERNAL
 neighbor 10.8.8.8 peer-group INTERNAL
 neighbor 10.9.9.9 peer-group INTERNAL
 neighbor 10.10.10.10 peer-group INTERNAL
!
ip prefix-list AGGREGATE seq 5 permit 142.55.0.0/16
!
```

```
route-map TO_TORIX permit 10
 match ip address prefix-list AGGREGATE
!
route-map TO_TORIX permit 20
 match as-path 23
!
route-map TO_TORIX permit 30
 match as-path 36
 set metric 10
!
route-map TO_TORIX deny 40
!
ip as-path access-list 23 permit _23457$
ip as-path access-list 36 permit _36750$
```

```
route-map TO_COGENT permit 10
 match ip address prefix-list AGGREGATE
!
route-map TO_COGENT deny 40
!
```

```
route-map SHERIDAN_iBGP deny 10
 match as-path 56
!
route-map SHERIDAN_iBGP permit 20
!
ip as-path access-list 56 permit ^$
```

Figure 10.26

Router R7 advertises the aggregate 142.55.0.0/16 to both TorIX and Cogent. Using network commands, any of the subnets that might be present in the routing table will keep the aggregation advertisement activated for it is unlikely that the four subnets will fail at the same time.

R7 advertises the routes 142.55.0.0/16 from Sheridan and 192.82.150.0/24 from Brampton with a metric lower than the metric of 10 advertised by router R6. If IP traffic arrives to TorIX, destined to either Sheridan or Brampton, then such traffic will flow via R7. If the link between R7 and R3 were to fail, then the same traffic will have no choice but to flow via R6. That is the intention of the policy to have a preferential path but also to have a fail-protection path.

On the other hand, R7 advertises the route from the City of Mississauga with a higher metric (10) than the metric advertised by R6. In consequence, the traffic coming from TorIX destined to 142.240.254.0/24 will flow via the link between R2 and R6. In case of a link failure, there is a second best path, between R3 and R7, waiting to be promoted to the routing table.

R7 advertises Sheridan's aggregation and nothing else to Cogent. When multiple links are available between two routers, BGP chooses one link to be the best path and the others remain in stand-by in the BGP database. Then BGP presents the best path to IP for its inclusion in the routing table. In fact, BGP does not do load balancing by default. In this current case, this behaviour of BGP is not desirable. If a customer is going to pay for two high speed links, they better be used. The command "maximum path" allows the inclusion of more than one link into the BGP best-paths so they can be used for load balancing. (note: the implementation of this command varies with the Cisco IOS versions.)

The remainder of the configuration deals with the internal BGP which is the same policy that has been implemented in all of Sheridan's routers. Router R7 re-advertises the routing information received from Cogent and Orion. Since there is no specific inbound policy, these routes will be marked with the default Local_Preference value of 100. This value is more attractive than the lower value (66) set by router R6. Hence R7 will be preferred as the exit point for the whole AS 5664.

Router R8 in AS 5664

This is a simple configuration router. Externally, it only peers with Cogent to advertise Sheridan's aggregation alone. However, Sheridan would like that the link between R8 and R4 be used as a last resort in case that the other links fail. So for that reason the as-path 5664 is prepended to make the AS_PATH to appear longer hence less desirable.

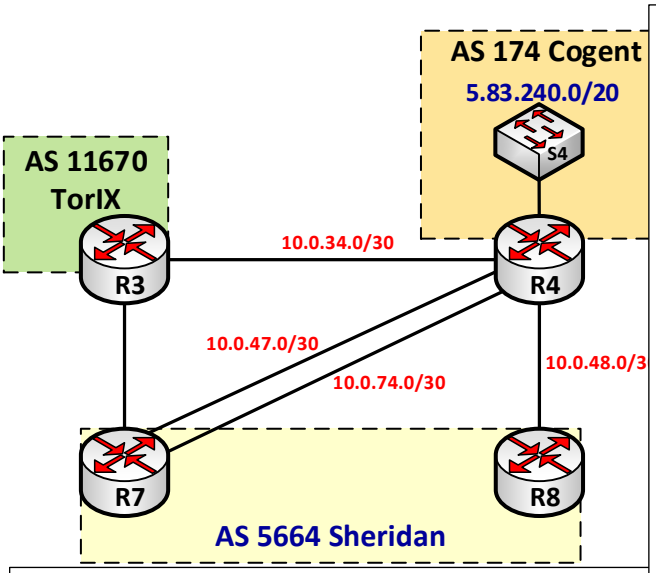
```
router bgp 5664
network 142.55.128.0 mask 255.255.192.0
 network 142.55.192.0 mask 255.255.192.0
 aggregate-address 142.55.0.0 255.255.0.0

 neighbor COGENT peer-group
 neighbor COGENT remote-as 174
 neighbor COGENT route-map TO_COGENT out
 neighbor 10.0.48.1 peer-group COGENT

neighbor INTERNAL peer-group
 neighbor INTERNAL remote-as 5664
 neighbor INTERNAL update-source Loopback0
 neighbor INTERNAL next-hop-self
 neighbor INTERNAL route-map SHERIDAN_iBGP out
 neighbor 10.6.6.6 peer-group INTERNAL
 neighbor 10.7.7.7 peer-group INTERNAL
 neighbor 10.9.9.9 peer-group INTERNAL
 neighbor 10.10.10.10 peer-group INTERNAL
!
ip as-path access-list 56 permit ^$
!
ip prefix-list AGGREGATE seq 5 permit 142.55.0.0/16
```

```
route-map TO_COGENT permit 10
 description "advertise 142.55/16 to Cogent"
 match ip address prefix-list AGGREGATE
set as-path prepend 5664
!
```

```
route-map SHERIDAN_iBGP deny 10
 match as-path 56
!
route-map SHERIDAN_iBGP permit 20
!
```

Cogent AS 174

Spanning throughout three continents, in more than 40 countries, with over 5,000 peers, Cogent [2] is the most interconnected Internet Service Provider. Connecting to AS 174 practically means being one AS hop count away to the most important routing domains in the Internet. In real life, Sheridan College, Brock University, and Orion all peer with Cogent, but representing all these connections would make the diagram too complex for this explanation. Beside the topic is the enterprise network. Therefore, Cogent is just shown connecting with Sheridan and TorIX.




```
router bgp 174
 network 5.83.240.0 mask 255.255.240.0

neighbor SHERIDAN peer-group
neighbor SHERIDAN remote-as 5664
neighbor 10.0.47.2 peer-group SHERIDAN
neighbor 10.0.74.2 peer-group SHERIDAN
neighbor 10.0.48.2 peer-group SHERIDAN
!
neighbor 10.0.47.2 route-map CLIENT in
neighbor 10.0.48.2 route-map CLIENT in
neighbor 10.0.74.2 route-map CLIENT in
!
maximum-paths 3
!
neighbor TORIX peer-group
neighbor TORIX remote-as 11670
neighbor 10.0.34.1 peer-group TORIX
```

```
route-map CLIENT permit 10
 match ip address prefix-list NO-MARTIAN
!
ip prefix-list NO-MARTIAN seq 5 deny 0.0.0.0/8 le 32
ip prefix-list NO-MARTIAN seq 10 deny 0.0.0.0/0
ip prefix-list NO-MARTIAN seq 15 deny 10.0.0.0/8 le 32
ip prefix-list NO-MARTIAN seq 20 deny 127.0.0.0/8 le 32
ip prefix-list NO-MARTIAN seq 25 deny 169.254.0.0/16 le 32
ip prefix-list NO-MARTIAN seq 30 deny 172.16.0.0/12 le 32
ip prefix-list NO-MARTIAN seq 35 deny 192.0.2.0/24 le 32
ip prefix-list NO-MARTIAN seq 40 deny 192.168.0.0/16 le 32
ip prefix-list NO-MARTIAN seq 45 deny 224.0.0.0/4 le 32
ip prefix-list NO-MARTIAN seq 50 deny 240.0.0.0/4 le 32
ip prefix-list NO-MARTIAN seq 55 permit 0.0.0.0/0 le 32
```

Figure 10.27

Verification

The whole topology is now completed. It is time to verify the results. Router R1 at Orion's AS 26677 has the following BGP knowledge (colour blue is being used for the best path):

R1-Orion#show ip bgp (edited)

Network	Next Hop	Metric	AS_Path	Organizations
*> 5.83.240.0/20	10.0.12.2		11670 174 i	TorIX Cogent
*> 38.112.112.0/24	0.0.0.0	0	i	Orion (internal)
*> 139.57.0.0	10.0.15.2	0	16475 i	Brock University
*> 142.55.0.0	10.0.16.2	0	5664 i	Sheridan
*	10.0.12.2		11670 5664 i	TorIX Sheridan
*> 142.240.254.0/24	10.0.16.2		5664 36750 i	Sheridan Mississauga
*	10.0.12.2		11670 5664 36750 i	TorIX Sheridan Mississauga
*> 192.82.150.0	10.0.16.2		5664 23457 i	Sheridan Brampton
*	10.0.12.2		11670 5664 23457 i	TorIX Sheridan Brampton

The best path information is graphically represented next:

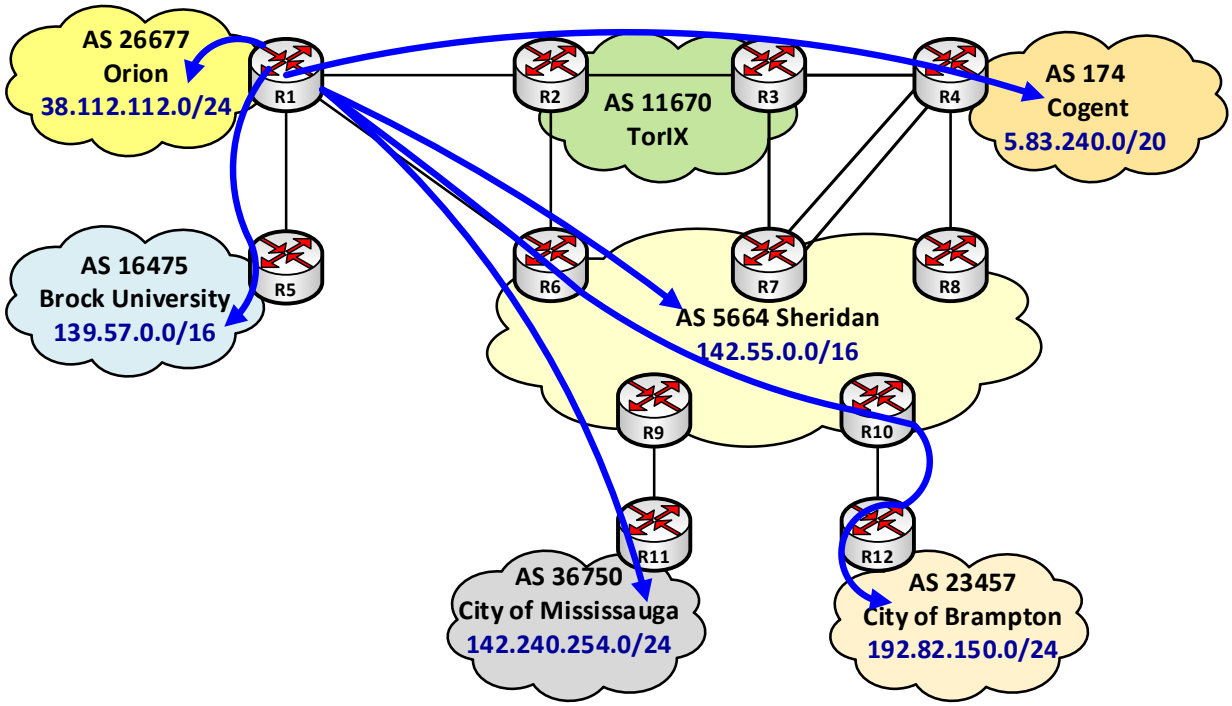


Figure 10.28

Which is all correct. Orion can see the route from Brock and it can see that routes from Sheridan and the two cities to be reachable preferably via R6, just as the routing policies meant to do.

R2 in TorIX has the following information in its BGP database:

R2-TorIX#show ip bgp (edited)

Network	Next Hop	Metric	LocPrf	Path	Organizations
*>i 5.83.240.0/20	10.0.23.2	0	100	174 i	Cogent
*> 38.112.112.0/24	10.0.12.1	0		26677 i	Orion
*>i 142.55.0.0/16	10.0.23.2	0	100	5664 i	Sheridan College
*	10.0.26.2	10		5664 i	Sheridan College
*>i 142.240.254.0/24	10.0.26.2			5664 36750 i	Sheridan City of Mississauga
*>i 192.82.150.0	10.0.23.2	0	100	5664 23457 i	Sheridan City of Brampton
*	10.0.26.2	10		5664 23457 i	Sheridan City of Brampton

The same information graphically represented:

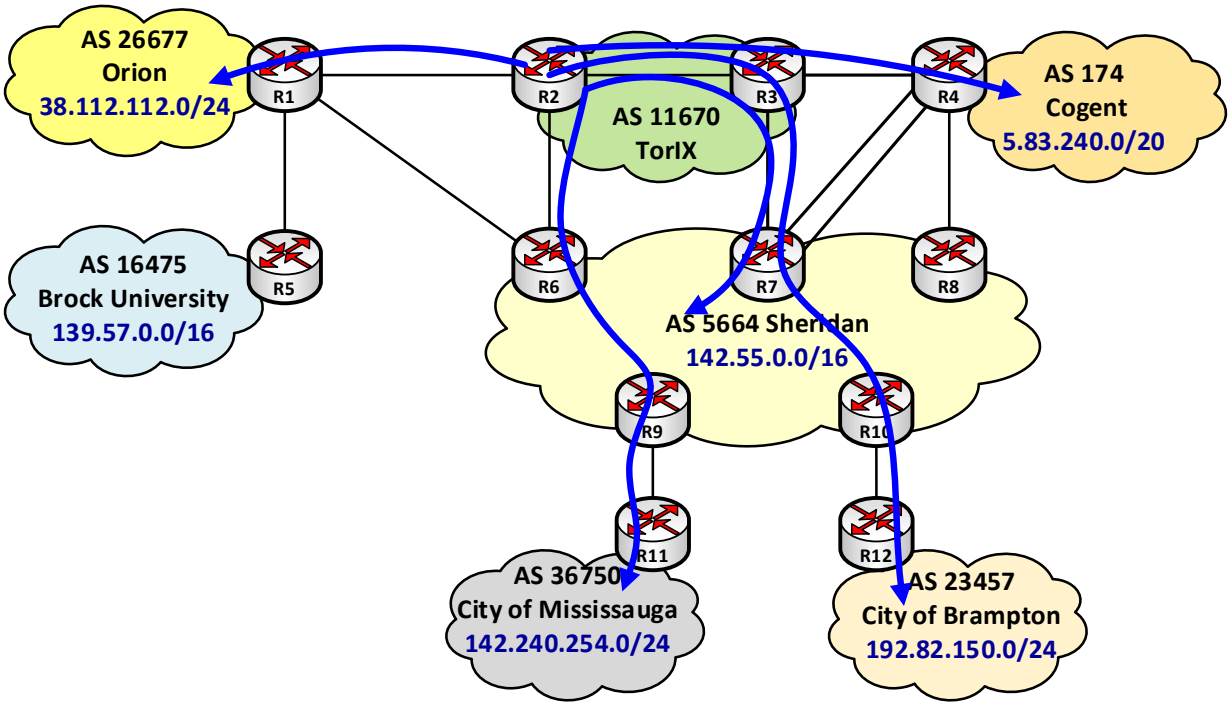


Figure 10.29

- R2 reaches Orion's route, but not Brock's since that is not advertised to TorIX.
- R2 sees that the Mississauga's route best path is via R6 because of the lower metric.
- The best path to both Sheridan's aggregation and Brampton's route is via R3 because of the lower metric advertised by R7.
- Finally, R2 has visibility of Cogent's route with R3 as the next hop router.
- The best paths from R2 comply with the intentions of the policies.

Router R3 in TorIX has the following routing information in its BGP database:

R3-TorIX#show ip bgp (edited)

Network	Next Hop	Metric	LocPrf	Path	Organizations
*> 5.83.240.0/20	10.0.34.2	0		174 i	Cogent
*>i 38.112.112.0/24	10.0.23.1	0	100	26677 i	Orion
* 142.55.0.0/16	10.0.34.2			174 5664 i	Cogent Sheridan
*>	10.0.37.2	0		5664 i	Sheridan
*> 142.240.254.0/24	10.0.23.1	0	100	5664 36750 i	Sheridan City-Mississauga
*	10.0.37.2	10		5664 36750 i	Sheridan City-Mississauga
*> 192.82.150.0/24	10.0.37.2			5664 23457 i	Sheridan City-Brampton

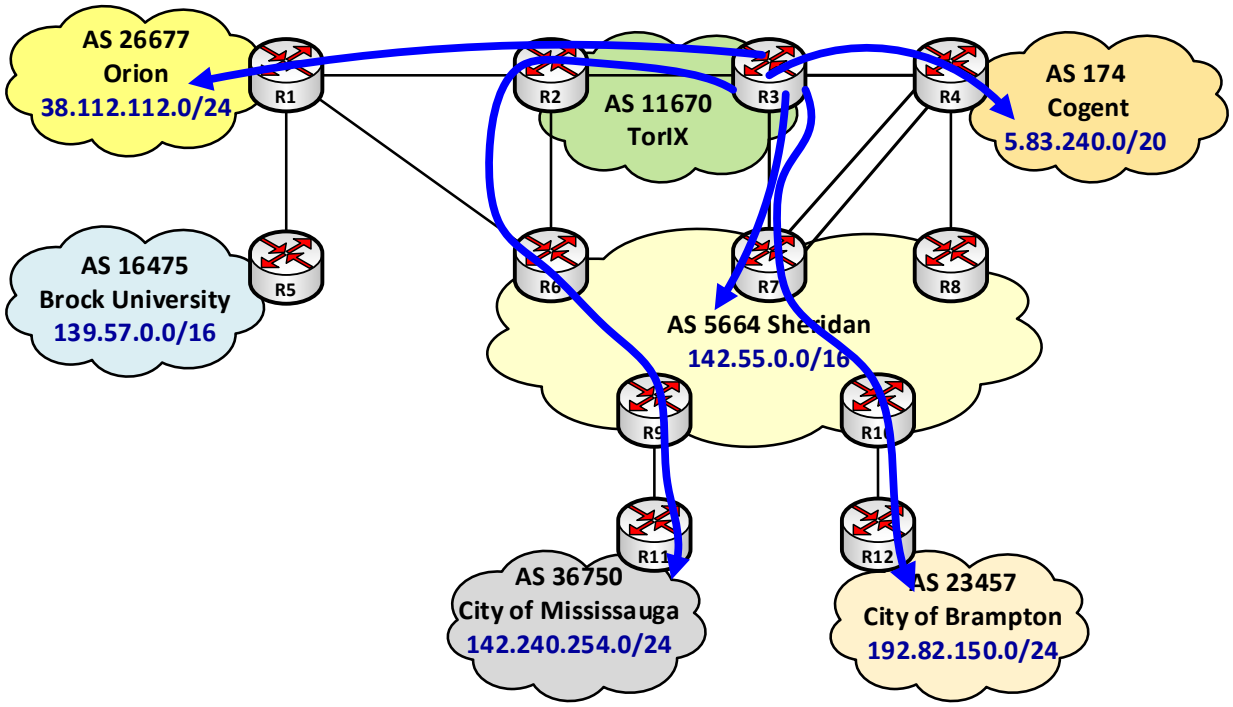


Figure 10.30

- The best path to Sheridan's aggregate is via R7 because of the lower metric.
- R3 has another path to the same aggregation via Cogent but that is not the shortest AS_PATH.
- R3 best path to the City of Mississauga is via R2 because of the lower metric advertised by R6.
- There is only one possible path to Brampton because R2 does not offer a different path.
- The rest of the information further verifies that the policies in place are acting correctly.

Router R4, located in Cogent, knows the following BGP information:

R4-Cogent#show ip bgp (edited)

Network	Next Hop	Metric	Path	Organizations
*> 5.83.240.0/20	0.0.0.0	0	i	Cogent (internal)
*> 38.112.112.0/24	10.0.34.1		11670 26677 i	TorIX Orion
* **142.55.0.0/16**	10.0.48.2	0	5664 5664 i	Sheridan (prepended)
*	10.0.34.1		11670 5664 I	TorIX Sheridan
*m	10.0.74.2	0	5664 i	Sheridan
*>	10.0.47.2	0	5664 i	Sheridan
*> 142.240.254.0/24	10.0.34.1		11670 5664 36750 i	TorIX Sheridan Mississauga
*> 192.82.150.0/24	10.0.34.1		11670 5664 23457 i	TorIX Sheridan Brampton

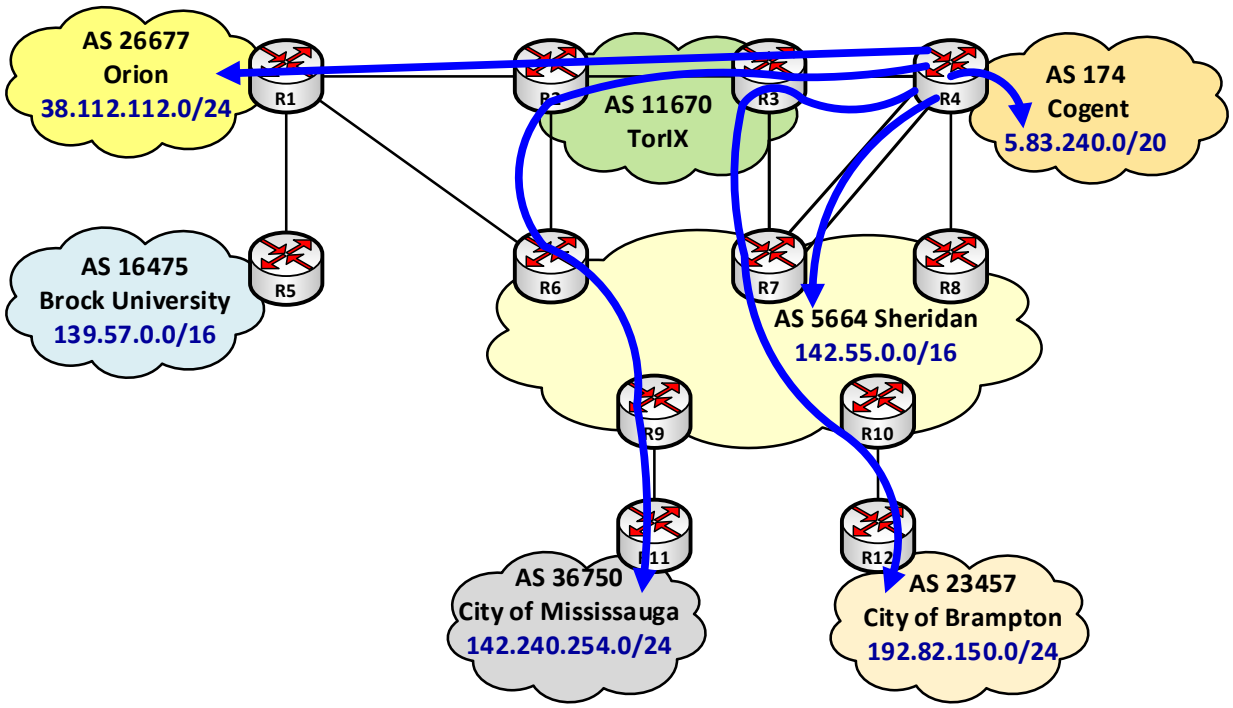


Figure 10.31

- It is worth to emphasize that R4 has four paths to Sheridan's aggregation 142.55.0.0/16.
- The first path to the aggregation goes via R8 but that is not the best one, since it has AS_PATH prepending. This proves that the policy at R8 is working as intended. For this path to be used, all the other better paths would have to fail.
- The second valid path to 142.55.0.0/16 is via TorIX and then Sheridan. This happens because TorIX learns the route from R6 and R7 and then it advertises its best path to Cogent. However, is not the best path for Cogent due to the longer AS_PATH.

- The dual links 10.0.47.0/30 and 10.0.74.0/30 between R4 and R7 are both counted as best paths as intended by policy. The letter **m** indicates that multipath has been enabled. Further verification of the multipath behaviour is shown with the next snippet. The IP routing table shows that there are two equally good paths to be used for load balancing to 142.55.0.0/16 from the viewpoint of Cogent.

```
R4-Cogent#show ip route bgp

B    142.55.0.0/16        [20/0] via 10.0.74.2
                          [20/0] via 10.0.47.2
B     38.112.112.0        [20/0] via 10.0.34.1
B     142.240.254.0       [20/0] via 10.0.34.1
B    192.82.150.0/24      [20/0] via 10.0.34.1
```

These two paths are present in the IP routing table for prefix 142.55.0.0/16.

- Notice that the remaining valid paths to the aggregation which are present in the BGP database table, are not present in the IP routing table. They were not promoted.
- Beside that, Cogent's R4 can send traffic to Orion's network via the transit path offered by TorIX. But Sheridan does not offer transit, as it should not, to Cogent to get to Orion. So polices are correct.
- Sheridan does offer a transit path to the City of Mississauga passing thru TorIX and then entering R6. This results from a combination of policies in different routers.
- Sheridan also offer a path to the City of Brampton via TorIX only.

R5 in Brock University has the following BGP database:

R5-Brock#show ip bgp (edited)

Network	Next Hop	Metric	Path	Organizations
*> 38.112.112.0/24	10.0.15.1	0	26677 i	Orion
*> 139.57.0.0	0.0.0.0	0	i	Brock University (internal)
*> 142.55.0.0	10.0.15.1		26677 5664 i	Orion Sheridan
*> 142.240.254.0/24	10.0.15.1		26677 5664 36750 i	Orion Sheridan Mississauga
*> 192.82.150.0	10.0.15.1		26677 5664 23457 i	Orion Sheridan Brampton

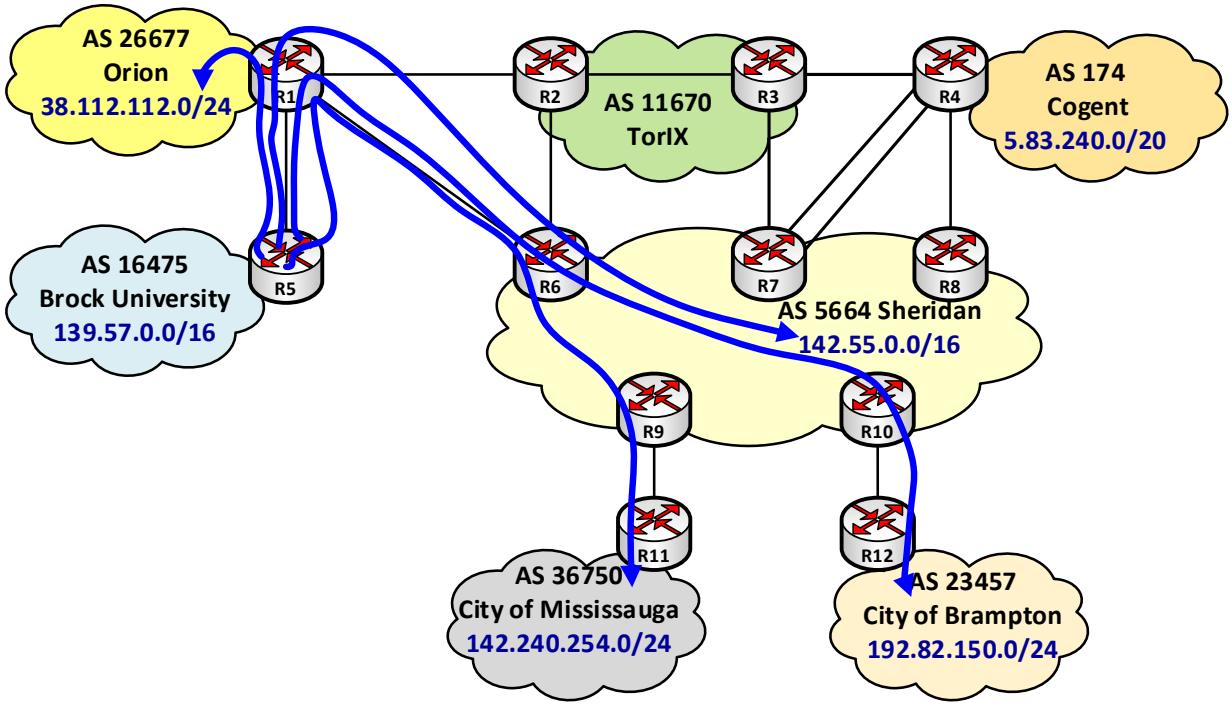


Figure 10.32

- R5, because of its position in the topology, obtains the knowledge resulting from the policies configured in R1, R6, and R2.
- R5 can reach Sheridan's aggregation and the Cities of Mississauga and Brampton thru Orion, then using R6 as a gateway.
- All these routes has community strings attached to it (this was verified earlier).
- R5 learns nothing about Cogent because Orion filters that knowledge out.

Router R6 located in Sheridan, has this BGP table:

R6-Sheridan#show ip bgp (edited)

Network	Next Hop	Metric	LocalPref	Path	Organizations
* 5.83.240.0/20	10.0.26.1		66	11670 174 i	TorIX Cogent
*>i	10.7.7.7	0	100	174 i	Cogent
* i	10.8.8.8	0	100	174 i	Cogent
* 38.112.112.0/24	10.0.26.1		66	11670 26677 i	TorIX Orion
*>	10.0.16.1	0	200	26677 i	Orion
*> 139.57.0.0	10.0.16.1		200	26677 16475 i	Orion Brock
*> 142.55.0.0/18	0.0.0.0	0		i	Sheridan (internal)
*> 142.55.0.0/16	0.0.0.0			i	Sheridan (internal)
*> 142.55.64.0/18	10.0.0.9	2		i	Sheridan (internal)
*>i 142.240.254.0/24	10.9.9.9	0	100	36750 i	Mississauga
*>i 192.82.150.0/24	10.10.10.10	0	100	23457 i	Brampton

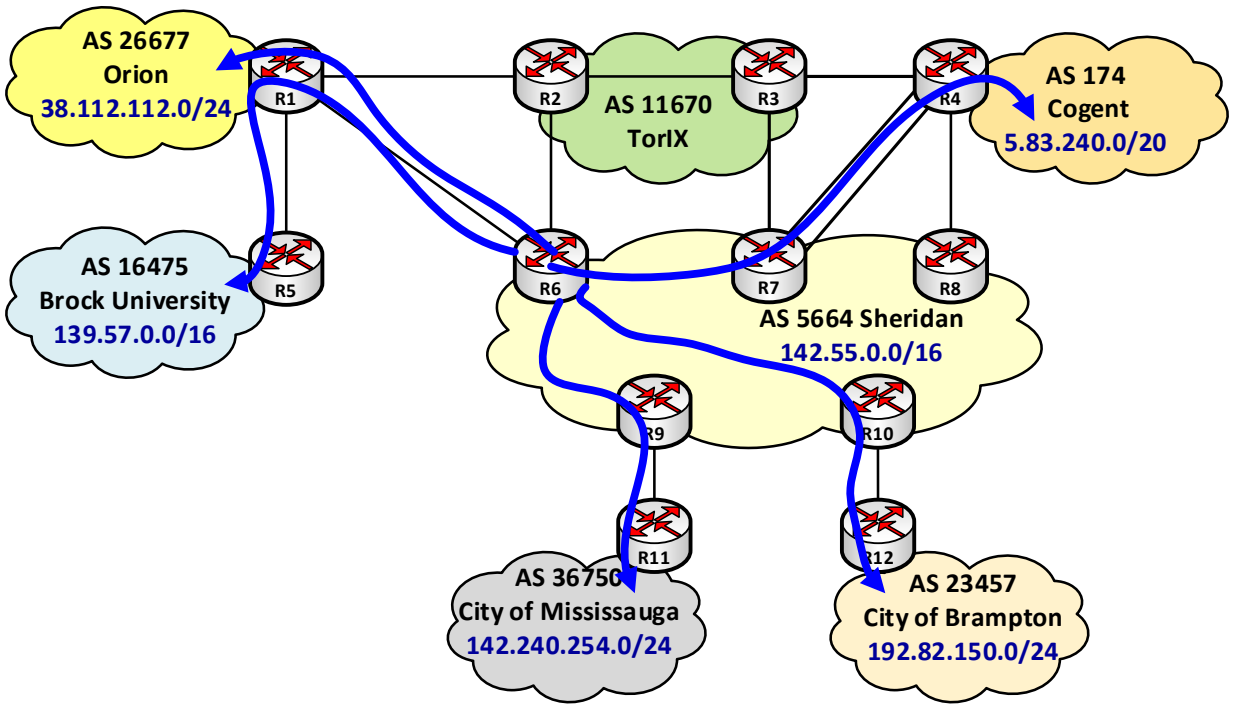


Figure 10.33

- R6 can see three valid paths to Cogent's 5.83.240.0/20. The path via TorIX has the longest AS_PATH, R6 marks that one with a low LOCAL_PREF of 66. The other routers in Sheridan would not see this value (66) unless that all better paths to 5.83.240.0/20 were to fail.

- R6 calculates that the best path to 5.83.240.0/20 is via 10.7.7.7 (R7) because the LOCAL_PREF of 100 and the shortest AS_PATH.

R6-Sheridan#show ip bgp (edited)

Network	Next Hop	Metric	LocalPref	Path	Organizations
* 5.83.240.0/20	10.0.26.1		66	11670 174 i	TorIX Cogent
*>i	10.7.7.7	0	100	174 i	Cogent
* i	10.8.8.8	0	100	174 i	Cogent

- There is another valid path to 5.83.240.0/20 via 10.8.8.8 (R8) which is not the best even though that it has the same LOCAL_PREF of 100 and the same AS_PATH length that the path via R7.
- It was intended that R7 would be the best exit gateway to Cogent, but this correct result was just luck. Basically, the BGP algorithm chose R7 because of the router-id.
- A more solid internal policy would be to have R7 advertising a higher LOCAL_PREF for 5.83.240.0/20 to the other Sheridan routers but excluding R8.
- R8 would be excluded so it can continue seeing its best path directly connected to Cogent instead of sending traffic internally to R7.
- Continuing with the analysis of R6's BGP table, this router finds a best path directly to Orion and a valid path doing transit thru TorIX. If the route with the best path is examined in detail, it would show that it is marked with the community 26677:20.
- R6 has only one path to Brock's route because Orion, by policy, does not advertises this client's route to TorIX. So TorIX cannot tell Sheridan about another potential path.
- R6 sets the LOCAL_PREF to 200 to the routes learned via Orion so all of Sheridan routers see this ASBR as the most preferred gateway to those destinations.
- R6 can see the internal paths to the cities of Mississauga and Brampton as advertised by routers R9 and R10.

Sheridan's R7 has an interesting BGP table:

R7-Sheridan#show ip bgp (edited)

Network	Next Hop	Metric	LocPrf	Path	Organization
* i 5.83.240.0/20	10.8.8.8	0	100	174 i	Cogent
*m	10.0.74.1	0		174 i	Cogent
*>	10.0.47.1	0		174 i	Cogent
*	10.0.37.1			11670 174 i	TorIX Cogent
* 38.112.112.0/24	10.0.74.1			174 11670 26677 i	Cogent TorIX Orion
*	10.0.47.1			174 11670 26677 i	Cogent TorIX Orion
*	10.0.37.1			11670 26677 i	TorIX Orion
*>i	10.6.6.6	0	200	26677 i	Orion
*>i 139.57.0.0	10.6.6.6	0	200	26677 16475 i	Orion Brock
*> 142.55.0.0/18	10.0.0.6	2		i	Sheridan (internal)
*> 142.55.0.0	0.0.0.0			i	Sheridan (internal)
*> 142.55.64.0/18	10.0.0.9	2		i	Sheridan (internal)
*> 142.55.128.0/18	10.0.0.8	2		i	Sheridan (internal)
*> 142.55.192.0/18	10.0.0.10	2		i	Sheridan (internal)
*>i 142.240.254.0/24	10.9.9.9	0	100	36750 i	Mississauga
*>i 192.82.150.0	10.10.10.10	0	100	23457 i	Brampton

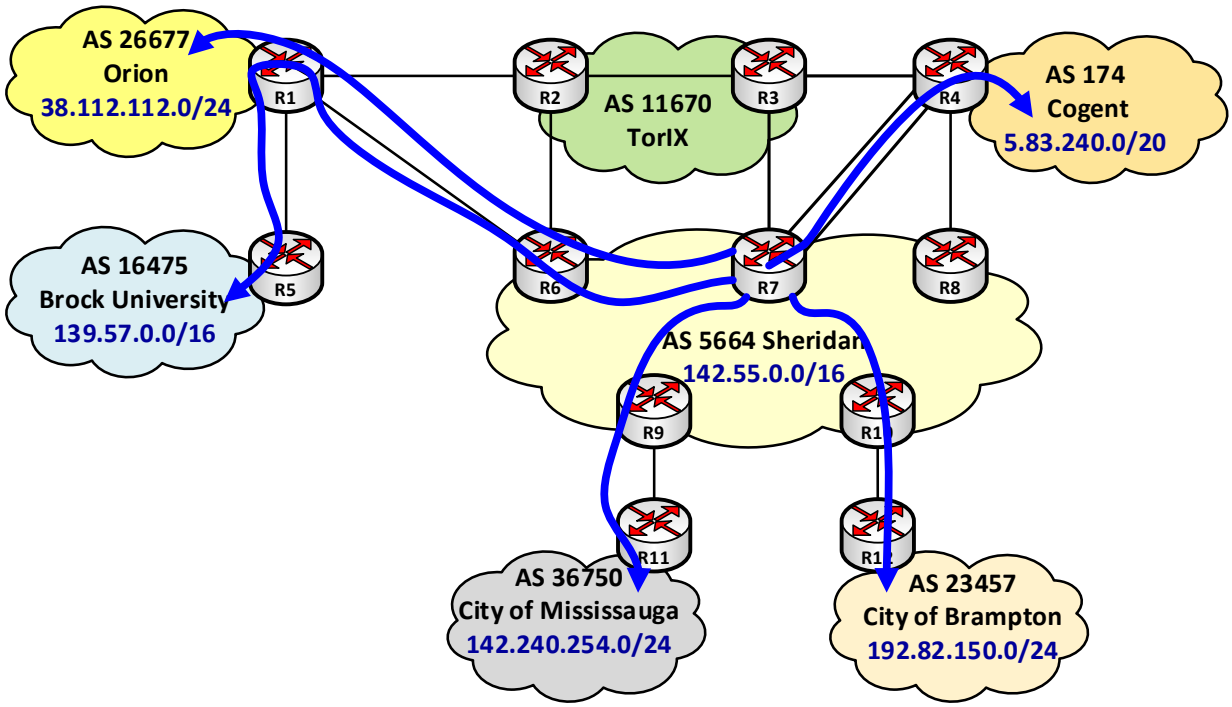


Figure 10.34

- R7 has four potential paths to Cogent's network 5.83.240.0/20. Although all are valid, just the two best paths are promoted to the routing table. The letter m indicates that the option for multiple paths have been enabled to do load balancing with the dual links between Sheridan and Cogent. Notice that the link between R8 and R4 is not marked with the m letter because the AS_PATH is longer.
- R7 can see four valid paths to get to Orion's 38.112.112.0/24 but the best one is via R6 because of the higher LOCAL_PREF value of 200 as intended by policy. If the best path were to fail, the second best would be promoted to the routing table and since the LOCAL_PREF values are all the same for the remaining three paths, the deciding factor would be the shortest AS_PATH via TorIX.
- There is only one path to Brock University route because it is only advertised to Sheridan's R6. The presence of this route proves that the policy in R6 is working properly because the match was done to the community 26677:10 to set a LOCAL_PREF of 200.
- The routes starting with the prefix 142.55 appear in the BGP table because of their inclusion with the network command.
- Finally, R7 can see what R9 and R10 advertise about the Cities of Mississauga and Brampton.

This is R8's BGP information:

R8-Sheridan#show ip bgp (edited)

Network	Next Hop	Metric	LocalPref	Path	Organizations
* i 5.83.240.0/20	10.7.7.7	0	100	174 i	Cogent
*>	10.0.48.1	0		174 i	Cogent
* 38.112.112.0/24	10.0.48.1			174 11670 26677 i	Cogent TorIX Orion
*>i	10.6.6.6	0	200	26677 i	Orion
*>i 139.57.0.0	10.6.6.6	0	200	26677 16475 i	Orion Brock
*> 142.55.0.0/16	0.0.0.0			i	Sheridan (internal)
*> 142.55.128.0/18	0.0.0.0	0		i	Sheridan (internal)
*> 142.55.192.0/18	10.0.0.10	2		i	Sheridan (internal)
*>i 142.240.254.0/24	10.9.9.9	0	100	i	City of Mississauga
*>i 192.82.150.0/24	10.10.10.10	0	100	23457 i	City of Brampton

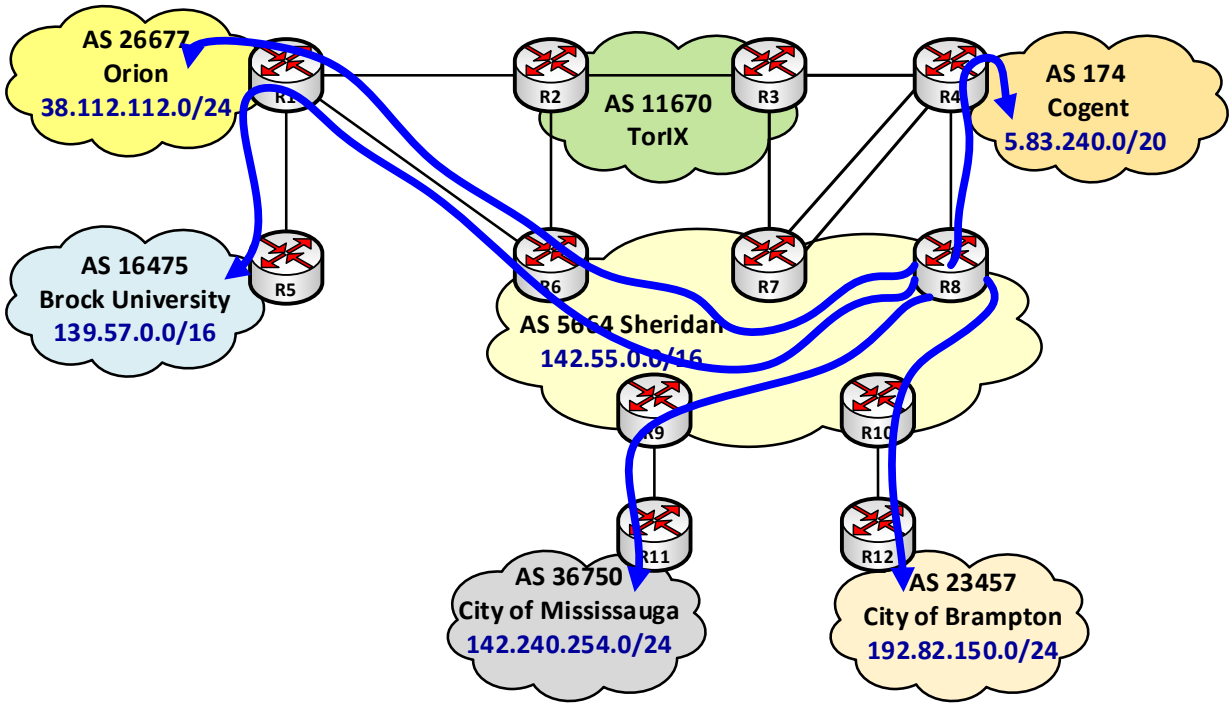


Figure 10.35

- R8 has a direct best path to Cogent's 5.83.240.0/20 which makes sense. However, if R7 were to advertise a better LOCAL_PREF, the path would switch toward R7. It is a design option to be considered. In that case, the link 10.0.48.0/30 would be idle mostly.
- R8 concludes that R6 is the preferred gateway to Orion and Brock because of the higher local preference attribute value (200).
- R8 also obtains the knowledge about the cities as advertised by R9 and R10.

Router R9 obtains this BGP base table:

R9-Sheridan#show ip bgp (edited)

Network	Next Hop	Metric	LocalPref	Path	Organizations
0.0.0.0	0.0.0.0			i	
*>i 5.83.240.0/20	10.7.7.7	0	100	174 i	Cogent
* i	10.8.8.8	0	100	174 i	Cogent
*>i 38.112.112.0/24	10.6.6.6	0	200	26677 i	Orion
*>i 139.57.0.0	10.6.6.6	0	200	26677 16475 i	Orion Brock
*> 142.240.254.0/24	10.0.119.2	0		36750 i	Mississauga
*>i 192.82.150.0	10.10.10.10	0	100	23457 i	Brampton

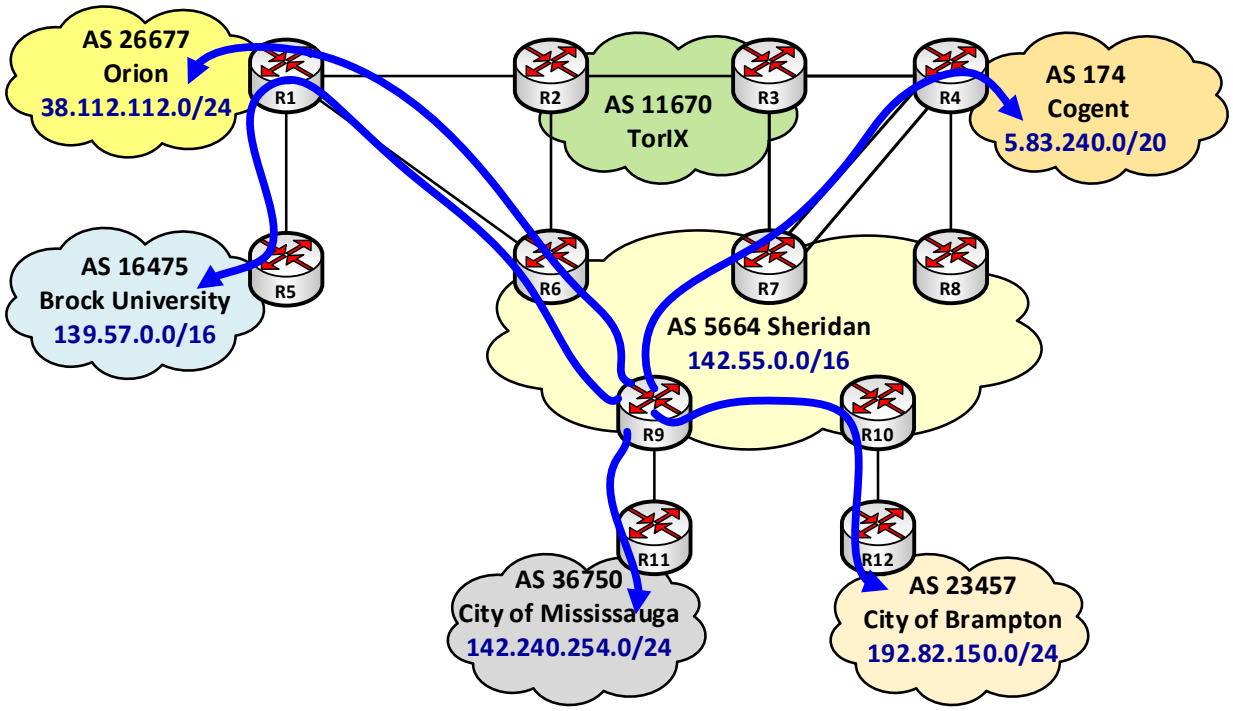


Figure 10.36

- The knowledge of this router depends of the best path decision already made by routers R6, R7 and R8. That is why there is only one path to each destination.
- R9 knows that R6 is the preferred ASBR to get to Orion and Brock (better LOCAL_PREF).
- R9 also knows that R7 is the preferred ASBR to get to Cogent but it knows that R8 also advertises a path to it. Although it was the design intention that R7 would be the exit gateway to go to Cogent, BGP has made the decision to choose R7 because of the lower router-id as compared to R8. This is a bit of a loose end that should not be left to chance.
- R9 learns 142.240.254.0/24 from Mississauga and the route from Brampton via R10.

R10 has a similar behaviour than R9.

R10-Sheridan#show ip bgp (edited)

Network	Next Hop	Metric	LocalPref	Path	Organizations
0.0.0.0	0.0.0.0			i	Sheridan (internal)
*>i 5.83.240.0/20	10.7.7.7	0	100	174 i	Cogent
* i	10.8.8.8	0	100	174 i	Cogent
*>i 38.112.112.0/24	10.6.6.6	0	200	26677 i	Orion
*>i 139.57.0.0	10.6.6.6	0	200	26677 16475 i	Orion Brock
*>i 142.240.254.0/24	10.9.9.9	0	100	36750 i	Mississauga
*> 192.82.150.0	10.0.112.2	0		23457 i	Brampton

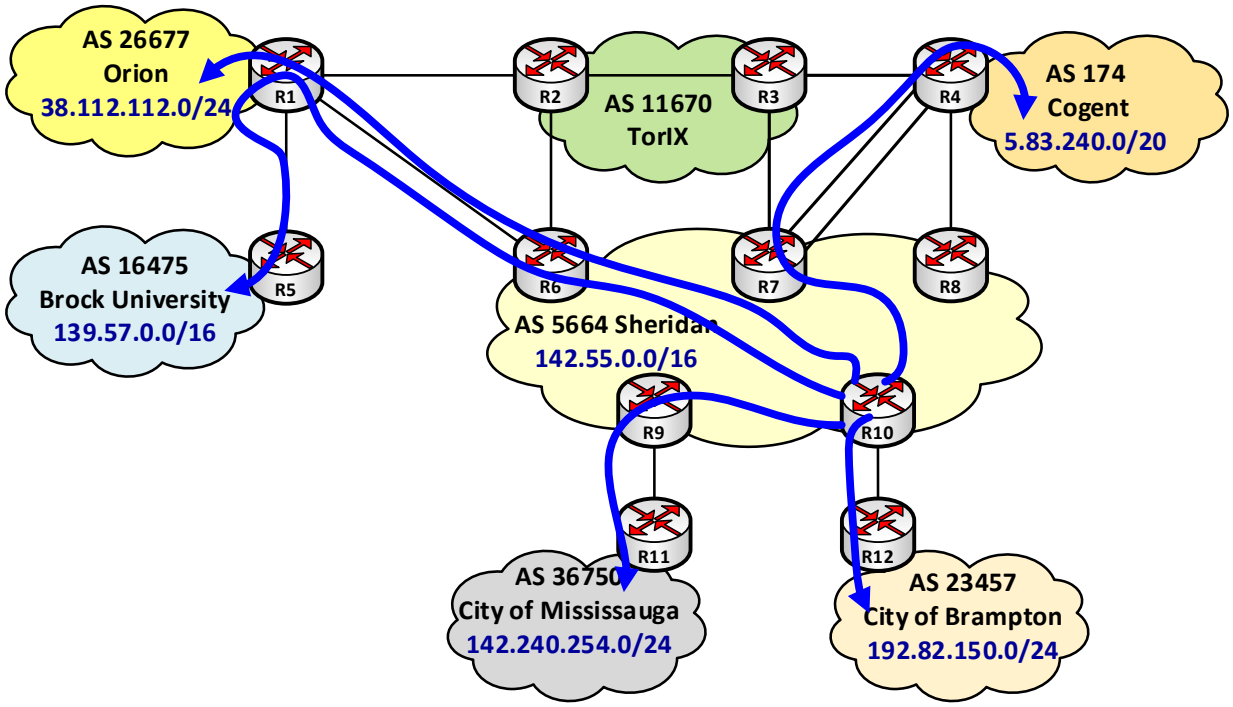


Figure 10.37

Chapter Summary

A made-up topology has been designed taking information from the public domain with the purpose of portraying an enterprise connecting to the Internet via ISP organizations. Several policies were designed and implemented to demonstrate different peering profiles.

The enterprise, AS 5664, privately collaborates with other similar organizations like AS 16475, AS 36750, and AS 23457. Then, the result of these routing agreements are not visible to the rest of the Internet. To do so, BGP policies were successfully used to correctly steer the routing advertisements.

A service provider, AS 26677, has a mission to facilitate connections among its members. But also, this organization provides data centre services to a broader audience. This provider defines several profile types with the use of BGP communities to deliver its services.

Finally, two major Internet Service Providers had the function of interconnecting AS 5664 to the simulated Internet at large. Since these ISP require very specific and complex configurations beyond of the scope of this chapter, they were just left with a bare bones setting.

Exercises

9.1 Repeat the configuration of figure 9.1 using Juniper routers only.

9.2 Implement the following fictitious case with both Cisco and Juniper routers:

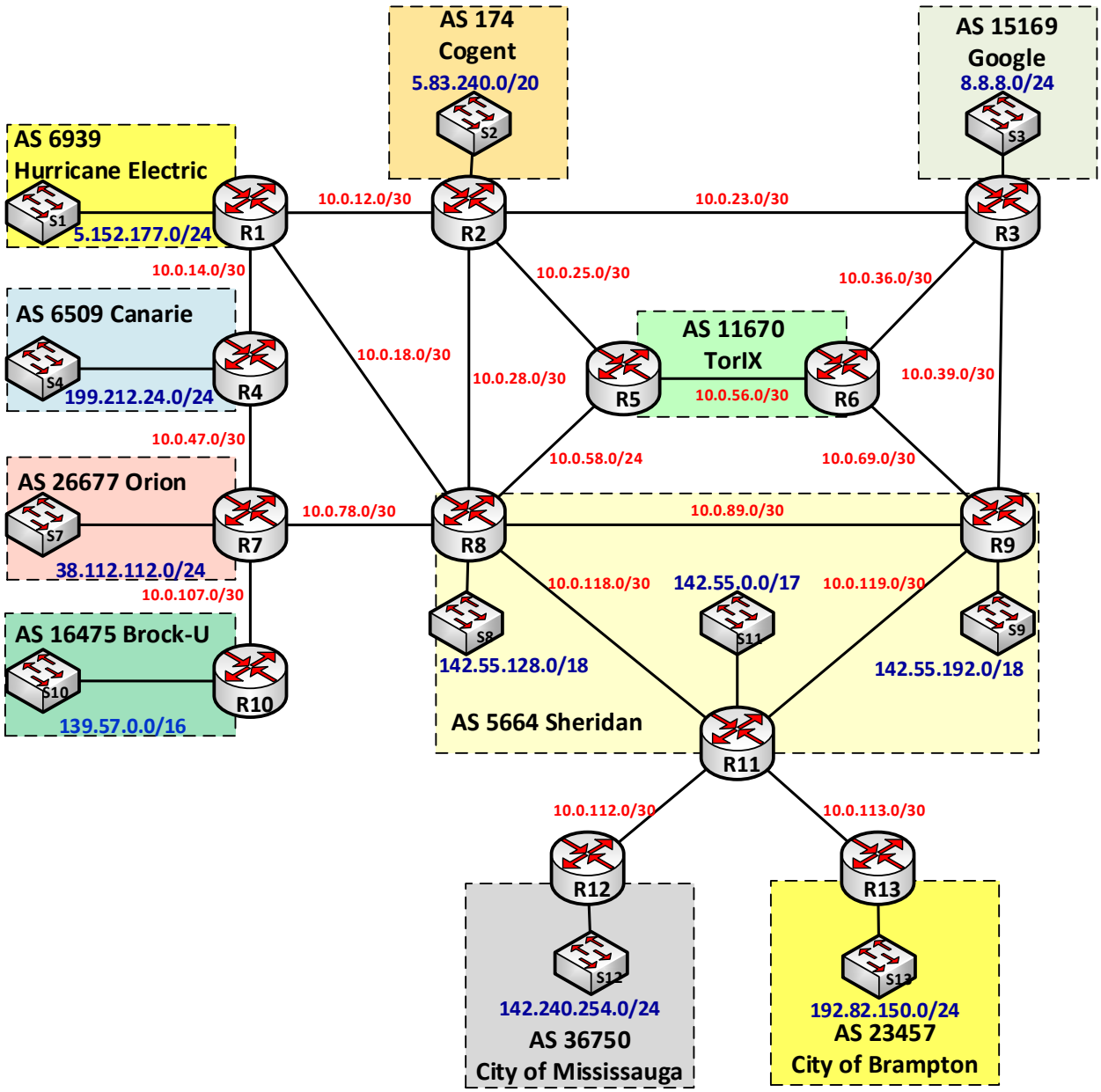


Figure 10.38

These are the routing intentions for the topology in figure 9.38.

- Sheridan advertises 142.55.192.0/18 to Google AS 15169.
- Google provides content to Sheridan, but in this case, Google is not providing access to other parts of the Internet. So Google does not re-advertise what it learns from Sheridan.
- Sheridan's R9 advertises the prefix from the City of Brampton to R6-TorIX.
- Sheridan's R8 advertises the prefix from the City of Mississauga to R5-TorIX.
- Sheridan advertises its aggregation 142.55.0.0/16 to TorIX.
- Sheridan's R8 advertises the aggregation 142.55.0.0/16 to Cogent marked with the community attribute 174:3002. Cogent will interpret this community [7] as "prepend AS174 twice" when re-advertising to other peers.
- Sheridan's R8 advertises the prefix 142.55.128.0/18 marked with the community string 174:3000. Cogent will interpret this community as [7] "do not announce to any peer".
- Sheridan's R8 advertises only the aggregation 142.55.0.0/16 to Hurricane Electric.
- Sheridan's R8 advertises only the aggregation 142.55.0.0/16 to Orion with the community Orion's member) 26677:10.
- Sheridan advertises to the Cities of Mississauga the following: Sheridan's aggregation, the Google's route 8.8.8.0/24 and anything that via Cogent (as a directly connected neighbour). Nothing else.
- Brock University advertises only its prefix with the community 26677:10 to Orion.
- Orion re-advertises routes marked with community 26677:10 to its members Sheridan and Brock. Also, it advertises these routes to Canarie.
- Orion advertises its local prefix with the community 26677:20 to Sheridan, Brock and Canarie.
- Canarie, does not re-advertises the prefixes from Sheridan and Brock to Hurricane Electric (HE).
- Canarie re-advertises the prefix from Orion to Hurricane Electric (HE).
- HE advertises its local prefix, Canarie's prefix and Orion's prefix to Cogent and to Sheridan.

Appendix

Case Topology and basic skeleton configuration

(*simulation done with GNS3)

* https://www.gns3.com

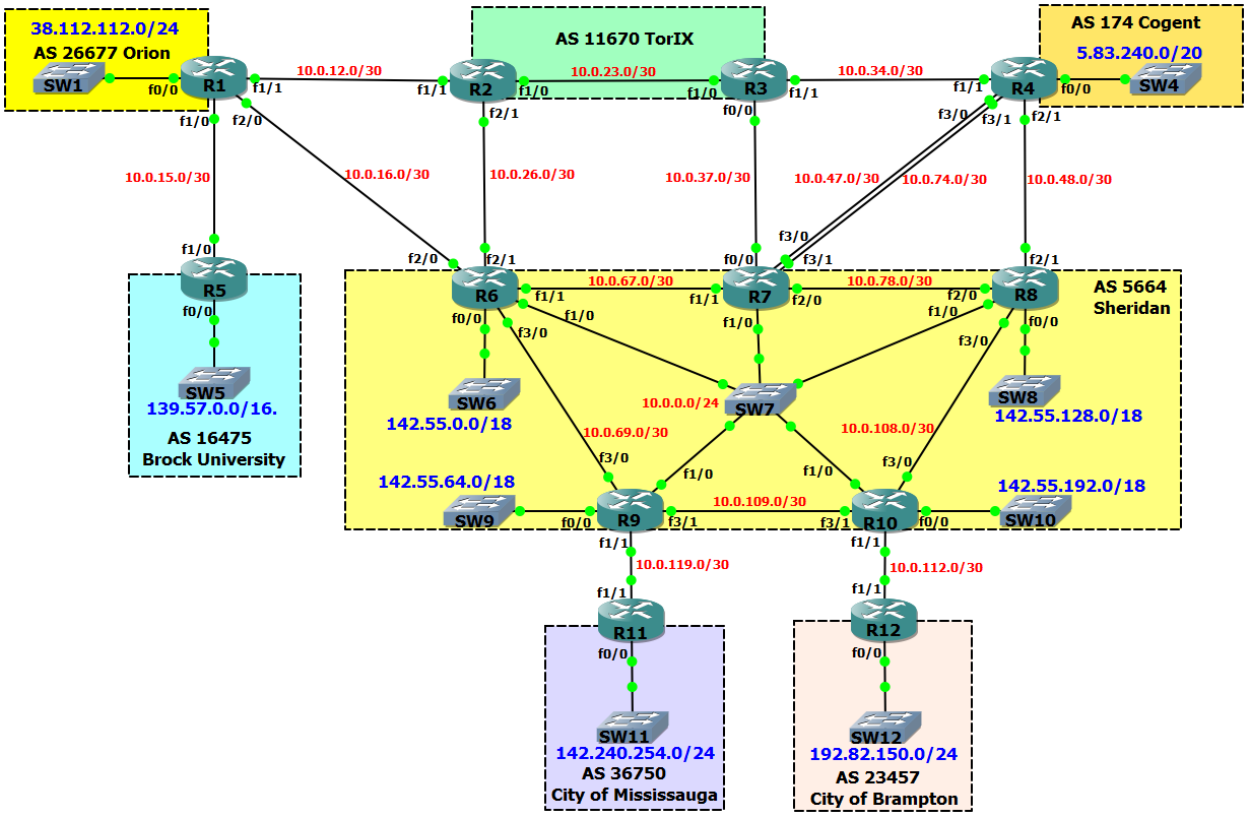


Figure 10.38

```
hostname R1-Orion
!
interface Loopback0
 ip address 10.1.1.1 255.255.255.255
!
interface FastEthernet0/0
 ip address 38.112.112.1 255.255.255.0
no shutdown
!
interface FastEthernet1/0
 ip address 10.0.15.1 255.255.255.252
 no shutdown
 !
interface FastEthernet1/1
 ip address 10.0.12.1 255.255.255.252
 no shutdown
 !
interface FastEthernet2/0
 ip address 10.0.16.1 255.255.255.252
no shutdown
```

```
hostname R2-TorIX
!
interface Loopback0
 ip address 10.2.2.2 255.255.255.255
!
interface FastEthernet1/0
 ip address 10.0.23.1 255.255.255.252
no shutdown
!
interface FastEthernet1/1
 ip address 10.0.12.2 255.255.255.252
 no shutdown
!
interface FastEthernet2/1
 ip address 10.0.26.1 255.255.255.252
no shutdown
!
router ospf 1
router-id 10.2.2.2
network 10.2.2.2 0.0.0.0 area 0
network 10.0.23.0 0.0.0.3 area 0
!
```

```
hostname R3-TorIX
!
interface Loopback0
 ip address 10.3.3.3 255.255.255.255
!
interface FastEthernet0/0
 ip address 10.0.37.1 255.255.255.252
no shutdown
!
interface FastEthernet1/0
 ip address 10.0.23.2 255.255.255.252
 no shutdown
!
interface FastEthernet1/1
 ip address 10.0.34.1 255.255.255.252
 no shutdown
!
router ospf 1
router-id 10.3.3.3
network 10.3.3.3 0.0.0.0 area 0
network 10.0.23.0 0.0.0.3 area 0
!
```

```
hostname R4-Cogent
!
interface Loopback0
 ip address 10.4.4.4 255.255.255.255
!
interface FastEthernet0/0
 ip address 5.83.240.1 255.255.240.0
no shutdown
!
interface FastEthernet1/1
 ip address 10.0.34.2 255.255.255.252
no shutdown
!
interface FastEthernet2/1
 ip address 10.0.48.1 255.255.255.252
no shutdown
!
interface FastEthernet3/0
 ip address 10.0.47.1 255.255.255.252
no shutdown
!
interface FastEthernet3/1
 ip address 10.0.74.1 255.255.255.252
 no shutdown
!
```

```
hostname R5-Brock
!
interface Loopback0
 ip address 10.5.5.5 255.255.255.255
!
interface FastEthernet0/0
 ip address 139.57.0.1 255.255.0.0
no shutdown
!
interface FastEthernet1/0
 ip address 10.0.15.2 255.255.255.252
no shutdown
!
```

```
hostname R6-Sheridan
!
interface Loopback0
ip address 10.6.6.6 255.255.255.255
!
interface FastEthernet0/0
 ip address 142.55.0.1 255.255.192.0
 no shutdown
!
interface FastEthernet1/0
 ip address 10.0.0.6 255.255.255.0
  no shutdown
 !
interface FastEthernet1/1
 ip address 10.0.67.1 255.255.255.252
  no shutdown
 !
interface FastEthernet2/0
 ip address 10.0.16.2 255.255.255.252
  no shutdown
 !
interface FastEthernet2/1
 ip address 10.0.26.2 255.255.255.252
  no shutdown
!
interface FastEthernet3/0
 ip address 10.0.69.1 255.255.255.252
  no shutdown
!
router ospf 1
 router-id 10.6.6.6
 network 10.0.0.0 0.0.0.255 area 0
 network 10.0.67.0 0.0.0.3 area 0
 network 10.0.69.0 0.0.0.3 area 0
 network 10.6.6.6 0.0.0.0 area 0
 network 142.55.0.0 0.0.63.255 area 0
!
```

```
hostname R7-Sheridan
!
interface Loopback0
 ip address 10.7.7.7 255.255.255.255
!
interface FastEthernet0/0
 ip address 10.0.37.2 255.255.255.252
!
interface FastEthernet1/0
 ip address 10.0.0.7 255.255.255.0
no shutdown
!
interface FastEthernet1/1
 ip address 10.0.67.2 255.255.255.252
no shutdown
!
interface FastEthernet2/0
 ip address 10.0.78.1 255.255.255.252
no shutdown
!
interface FastEthernet3/0
 ip address 10.0.47.2 255.255.255.252
no shutdown
!
interface FastEthernet3/1
 ip address 10.0.74.2 255.255.255.252
no shutdown
!
router ospf 1
 router-id 10.7.7.7
 network 10.0.0.0 0.0.0.255 area 0
 network 10.0.67.0 0.0.0.3 area 0
 network 10.0.78.0 0.0.0.3 area 0
 network 10.7.7.7 0.0.0.0 area 0
```

```
hostname R9-Sheridan
!
interface Loopback0
 ip address 10.9.9.9 255.255.255.255
!
interface FastEthernet0/0
 ip address 142.55.64.1 255.255.192.0
no shutdown
!
interface FastEthernet1/0
 ip address 10.0.0.9 255.255.255.0
no shutdown
!
interface FastEthernet1/1
 ip address 10.0.119.1 255.255.255.0
no shutdown
!
interface FastEthernet3/0
 ip address 10.0.69.2 255.255.255.252
no shutdown
 !
interface FastEthernet3/1
 ip address 10.0.109.1 255.255.255.252
no shutdown
 !
router ospf 1
 router-id 10.9.9.9
 network 10.0.0.0 0.0.0.255 area 0
 network 10.0.69.0 0.0.0.3 area 0
 network 10.0.109.0 0.0.0.3 area 0
 network 10.9.9.9 0.0.0.0 area 0
 network 142.55.64.0 0.0.63.255 area 0
```

```
hostname R10-Sheridan
!
interface Loopback0
 ip address 10.10.10.10 255.255.255.255
!
interface FastEthernet0/0
 ip address 142.55.192.1 255.255.192.0
no shutdown
!
interface FastEthernet1/0
 ip address 10.0.0.10 255.255.255.0
no shutdown
!
interface FastEthernet1/1
 ip address 10.0.112.1 255.255.255.0
no shutdown
!
interface FastEthernet3/0
 ip address 10.0.108.2 255.255.255.252
no shutdown
 !
interface FastEthernet3/1
 ip address 10.0.109.2 255.255.255.252
no shutdown
 !
router ospf 1
 router-id 10.10.10.10
 network 10.0.0.0 0.0.0.255 area 0
 network 10.0.108.0 0.0.0.3 area 0
 network 10.0.109.0 0.0.0.3 area 0
 network 10.10.10.10 0.0.0.0 area 0
 network 142.55.192.0 0.0.63.255 area 0
```

```
hostname R11-City of Mississauga
!
interface Loopback0
 ip address 10.11.11.11 255.255.255.255
!
interface FastEthernet0/0
 ip address 142.240.254.1 255.255.255.0
no shutdown
!
interface FastEthernet1/1
 ip address 10.0.119.2 255.255.255.0
no shutdown
!
```

```
hostname R12-City of Brampton
!
interface Loopback0
 ip address 10.12.12.12 255.255.255.255
!
interface FastEthernet0/0
 ip address 192.82.150.1 255.255.255.0
no shutdown
!
interface FastEthernet1/1
 ip address 10.0.112.2 255.255.255.0
no shutdown
!
```

Chapter References

[1] CIDR REPORT. (2016). http://www.cidr-report.org/as2.0/

[2] Cogent Communications. (2016). http://www.cogentco.com/en/

[3] Toronto Internet Exchange TorIX. (2016). https://www.torix.ca/

[4] Orion Research and Innovation Optical Network ORION. (2016). http://www.orion.on.ca/

[5] Shahid Ajaz. (2003) Orion BGP Communities.

http://www.orion.on.ca/wp-content/uploads/2012/06/ORION-BGP-Communities.pdf

[6] Team Cymru. (2016) http://www.team-cymru.org/bogon-reference.html

[7] Cogent Communications. (2012). Optical Internet, North America User Guide.

[8] Canarie (2016) http://www.canarie.ca

CHAPTER 11

BGP as a Signalling Protocol

BGP, with its flexible payload, can do more than just advertising IP routes. In fact, the protocol has found a number of applications whenever information of different nature needs to be propagated reliably. One of these applications is to carry information to interconnect remotely located sites to implement virtual networks. There are different formats or models based on this idea. In this chapter, the role of BGP in the model to provide layer VPN is analyzed and other models are conceptually outlined.

The functioning of a Service Providing Core

The carrier AS 65144 has the following physical topology. An internal gateway protocol (IGP) builds the internal IP connectivity. All routers have loopback addresses.

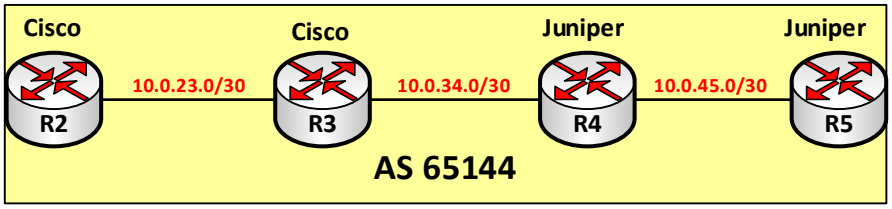


Figure 11.1

This is the philosophical principle of the functioning of this core. Let's assume that router R2 has "something" to send to router R5. The nature of this "something" is not relevant now as it will become clear when the specifics of the case are explained. What really matters is that both routers R2 and R5 know what to do with the "something".

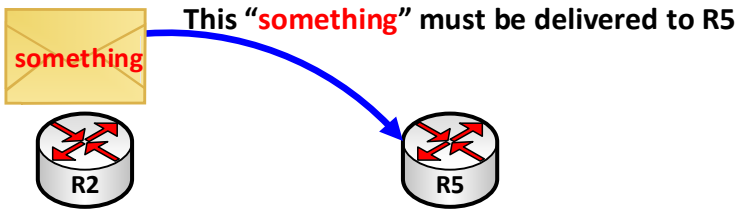


Figure 11.2

The big problem though is that the routers in the middle, R3 and R4, do not understand the "something" but they still have to deliver it somehow. The solution to this problem is that R2 places the "something" inside a "box" that all the routers understand. Basically, it is hidden inside an encapsulation.

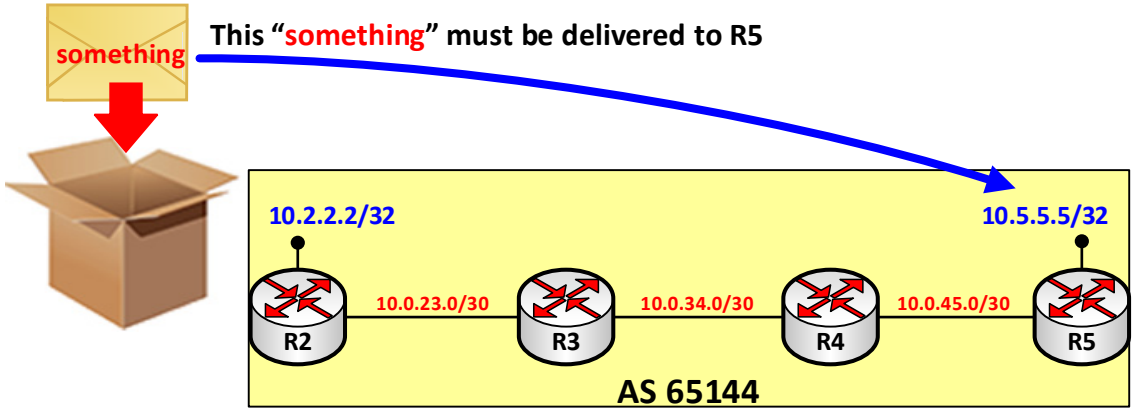


Figure 11.3

Previously, the routers has had a conversation, in a certain language, with their immediate neighbours and they had advertised "labels" to get to the destinations inside the core. For example, R3 told R2 the following: "for destination 10.5.5.5/32, place the sticker number 19 on the box". This talking is carried on with a language for label exchange specifically.

By combining the labels and the information from the IP routing table, the routers can build forwarding tables with labels pairing with internal IP destinations. For instance, R2 has learned from R3, that to send a "box" with "something" inside to destination 10.5.5.5, it has to label the box with a 19. That action is called **push** the label.

R2-PE#show mpls forwarding-table (edited)

Local-label	Outgoing-label	Prefix	Outgoing-interface	Next Hop (is R3)
16	Pop Label	10.3.3.3/32	f0/0	10.0.23.2
17	Pop Label	10.0.34.0/30	f0/0	10.0.23.2
20	17	10.4.4.4/32	f0/0	10.0.23.2
21	18	10.0.45.0/30	f0/0	10.0.23.2
22	19	10.5.5.5/32	f0/0	**10.0.23.2**

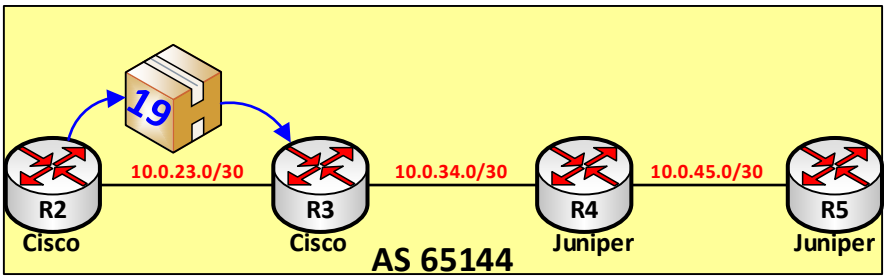


Figure 11.4

R3 receives the "box" with label 19 and it proceeds to check its own switching table. It knows that the destination 10.5.5.5/32 is located via next-hop 10.0.34.2 on interface f0/0. Then for a "box" received with label 19, router R3 **swaps** the label to the value that the next hop router (R4) is expecting. According to the table, R4 had advertised, with an LDP message, the label 299,808 for 10.5.5.5/32. So R3 sends the "box" marked with 299,808 toward R4.

R3-P#show mpls forwarding-table (edited)

Local-label	**Outgoing-label**	Prefix	Outgoing-interface	Next Hop
16	Pop Label	10.2.2.2/32	fa0/0	10.0.23.1
17	Pop Label	10.4.4.4/32	fa0/0	10.0.34.2
18	No Label	10.0.45.0/30	fa0/0	10.0.34.2
19	299808	10.5.5.5/32	fa0/0	**10.0.34.2**

R4 receives the "box" and then it proceeds to check its own switching table for the next action.

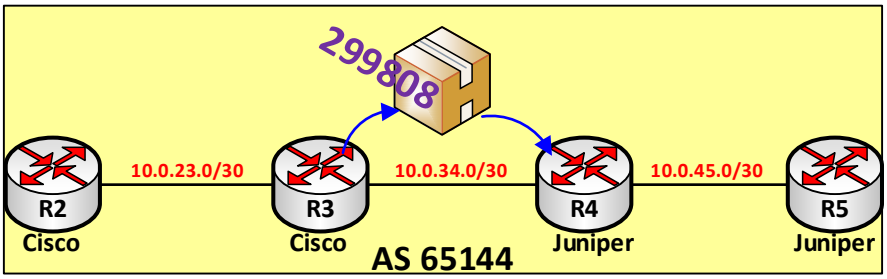


Figure 11.5

felix@R4> show route forwarding-table (edited)

Destination	**Next hop**	**Action**	**Net-if**
299776(S=0)	10.0.34.1	Pop	em1.0
299792	10.0.34.1	Swap 16	em1.0
299808 (S=0)	**10.0.45.2**	Pop	**em0.0**

R4 is a Juniper router so the table format is different, but anyways, it reads that the inbound "box" with the label 299,808 must be pop open and spill over what is inside (the "something") to the next hop 10.0.45.2 which is precisely the destination router R5:

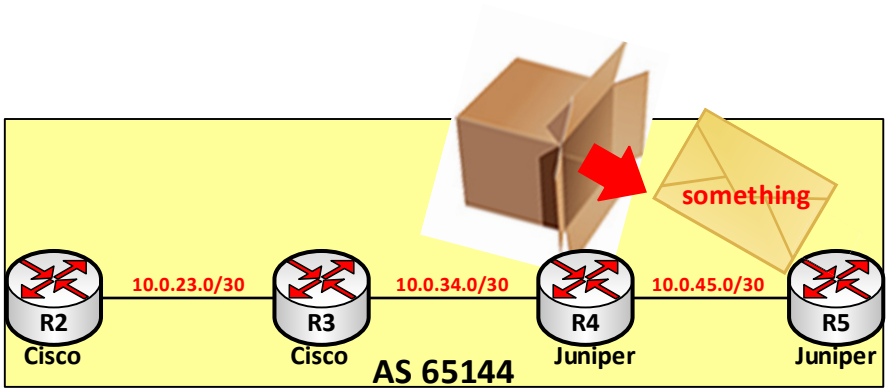


Figure 11.6

Then R5 has finally received the "something" and that is the whole purpose of this scheme. The routers inside the core (R3 and R4) have no idea what is inside the boxes. They just deliver the boxes by reading the labels outside. These routers are called **P routers**.

However, the routers at the edge of the core (R2 and R5) must have the knowledge to understand what to do with the inner contains of the boxes. These routers are called **PE routers** and they act as interfaces between the core and whatever is outside the core.

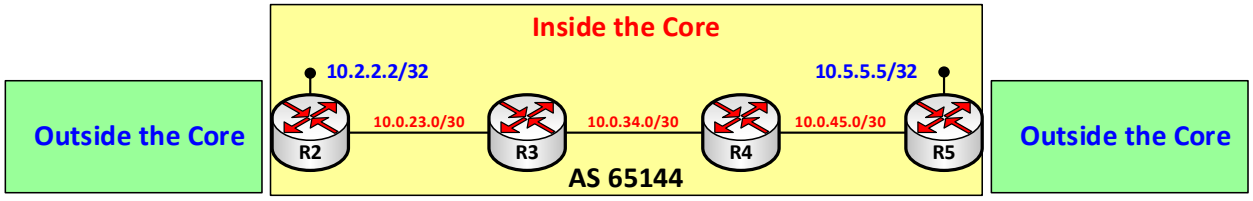


Figure 11.7

Consequently, the PE routers, R2 and R5, must have interfaces with an external view. Critically, these two routers must exchange these external information and what protocol is flexible enough to propagate information of different nature? That would be Border Gateway Protocol, in this case internal BGP. So R2-PE and R5-PE talk iBGP to each other

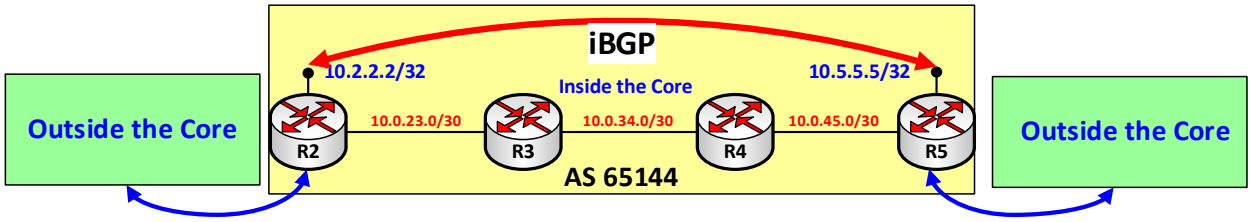


Figure 11.8

To summarize so far, inside the core AS 65144:

- An IGP routing protocol (OSPF or IS-IS) is needed to build the internal IP connectivity so all internal prefixes are reachable.

- One protocol maps the internal IP prefixes in the routing table to labels and advertises such labels to the directly connected neighbours. There are two protocols typically used for this function: **Label Distribution Protocol LDP** and **Reservation Protocol RSVP-TE**.

- A special protocol, pushes, swaps and pops encapsulation headers that have labels. That is **Multiprotocol Label Switching Protocol MPLS**.

- The border routers PEs talk to each other to communicate the external knowledge of things outside the core. They speak **iBGP** to do so.

When "something" arrives from the "outside" to a PE router with a remote destination. Such PE router puts the "something" into a "box" with a label and then it sends that to travel inside the core. At the end of the path there is another PE router that removes the "box" and delivers the internal contains. As a result, AS 65144 is a core that provides services to whatever is outside, basically customers.

To start completing the scenario, let's assume that "the outside" is an organization, called A, which has two locations, one in the city of Toronto and the other in Montreal. This company A would like to link the two sites in a transparent way with minimum configuration on its part as its technical resources and personnel are limited. Also, the company does not want to send its traffic over the global Internet. What is the solution for this problem?

Figure 11.9

Clearly, some infrastructure is needed between the two sites to make the interconnection possible. Precisely the solution has already been outlined with the previous core AS 65144, so it just a matter or placing that between the customer's sites.

This network provider now offers connectivity services to company A. From the viewpoint of AS 65144, traffic is received at the edges (what was called the "something" before) and it crosses the core inside encapsulating "boxes". From the viewpoint of the customer, the service provider looks just like a long pipe connecting the two locations. The customer only needs some basic configuration to get data flowing transparently from one side to the other. That simplicity makes this service attractive.

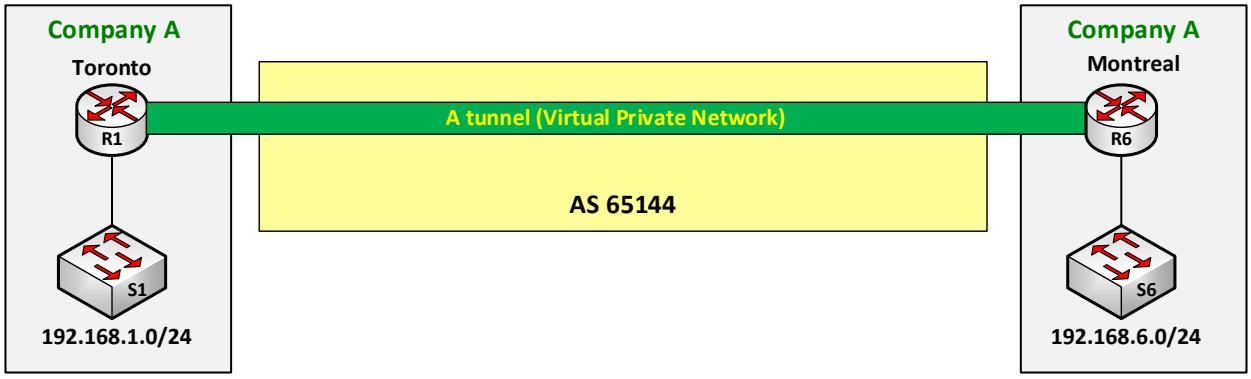


Figure 11.10

Until now, the traffic was being called "something" generically. In reality, the traffic received from the customer's sites could have different formats.

- It could be in the form of IPv4 packets.
- It could be in the form of IPv6 packets.
- It could be in the form of layer 2 frames (Ethernet or many others)
- It could be in other formats like an MPLS packet.

Each one of these formats implies a different provisioning model. When the traffic profile is based on IPv4 networks, the solution is called Layer 3 VPN IPv4 over MPLS (L3VPNv4 MPLS) and that is the focus of this explanation.

The model presented so far would work fine with just one customer. But obviously, that makes no sense for the service provider. Instead, the carrier would like to offer connectivity services to as many clients as possible while implementing standardized provisioning. Basically, a template that can be adapted to new customers. For example, another customer "company B" is added to the topology. This new customer has sites in the same cities than customer A and, just by lucky chance, it happens to have the same private IP addressing scheme.

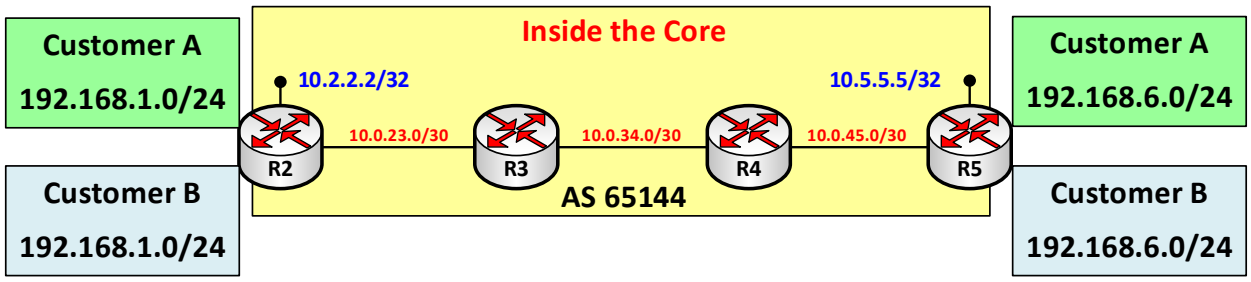


Figure 11.11

The main problem is not just that there is a conflict with the overlapping IP addressing. Even if the networks were renumbered, there is a still a bigger problem which is routing domain separation. The provider AS 65144 would never want to partake in the internal routing architecture of the customers. Furthermore, keeping the customers routing separated is a requirement. Hence, this model still needs one more component to make it truly functional.

In the PE routers, **virtual routing forwarding (VRF)** structures are created, one per customer. **VRF**s are **routing instances** that keep routing domains in isolation like in the following diagram where the PE router provides service to three customers A, B, and C.

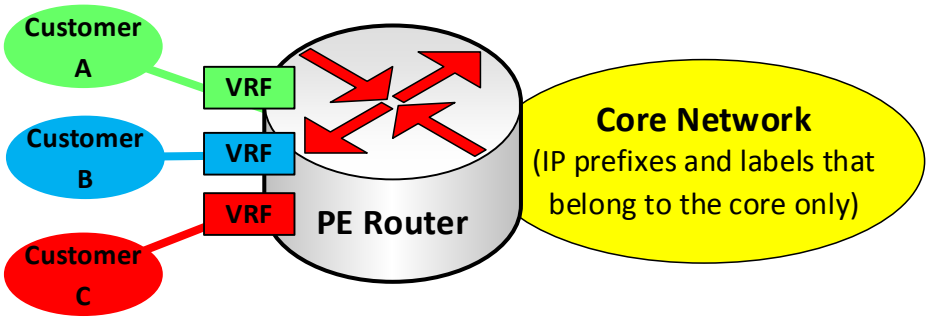


Figure 11.12

There are four (4) routing tables because there is the routing table of the core network which is also kept separated from the customers. So now, many customers with any kind of routing protocol can be provisioned and their traffic can be set apart. For instance, this network has symmetrical ends with customers green, blue and red.

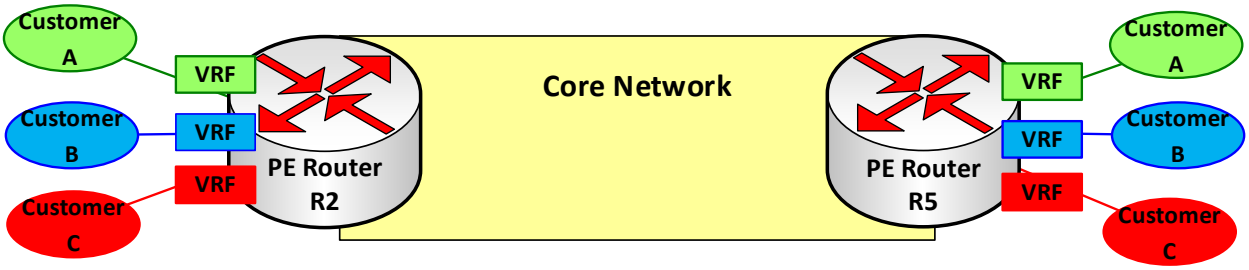


Figure 11.13

Every VRF must be configured on the PE router that interfaces with the customer. For example, the VRF for the customer green needs the following:

- An identification **name** for administration purposes.
- A unique identifier string. This is called the **route distinguisher** (RD).

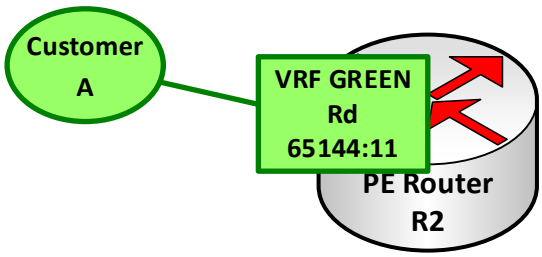


Figure 11.14

A common practice is to make the route-distinguisher with the autonomous system number (65144) plus a conventional number. In this case, 11 means site 1, customer 1. Consequently, the numbering for the three VRF would be something like this:

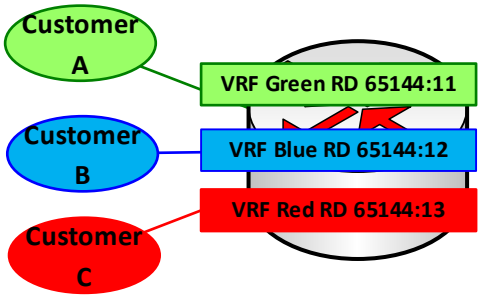


Figure 11.15

In the second location, a mirror configuration takes place.

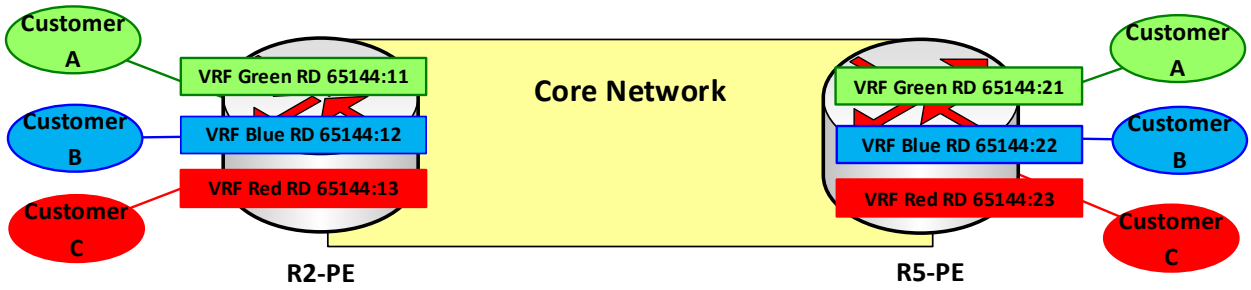


Figure 11.16

To understand the mechanism of the VRF, the green customer is selected for the explanation. The customer advertises the route prefix 192.168.2.0/24 with a routing protocol. It can be any routing protocol (OSPF, IS-IS, RIP, EIGRP, BGP, even static definition). The same routing protocol has to be spoken on the PE side. This routing knowledge gets installed in the green VRF table and it is isolated from any other routing table.

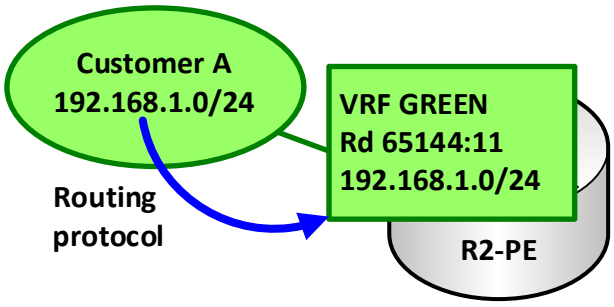


Figure 11.17

In location 2, the same process takes place. Prefix 192.168.6.0/24 is advertised to the PE router.

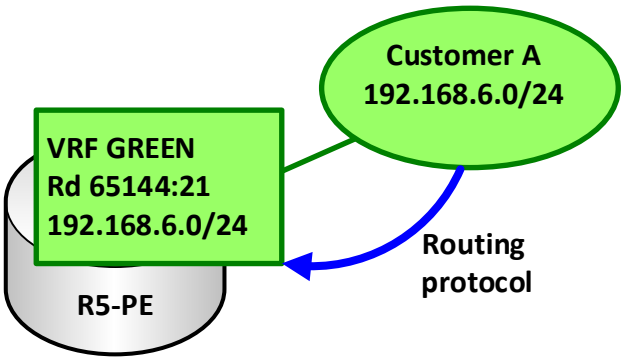


Figure 11.18

So the VRFs in both PE routers have the routing information from the clients. It is now a manner of exchanging such information so that traffic can be forwarded across the core. This is where BGP has a critical role. The border routers (PE) R2 and R5 exchange BGP UPDATE messages containing the following:

- The IPv4 prefixes with their masks.
- The route-distinguishers.
- A label for the prefix.

It is to notice that the PE routers push a label for the prefix that is going to be advertised and that these labels are completely unrelated to the labels inside the core. The process of BGP information exchange is portrayed in the next diagram. (label numbers were made up for the explanation)

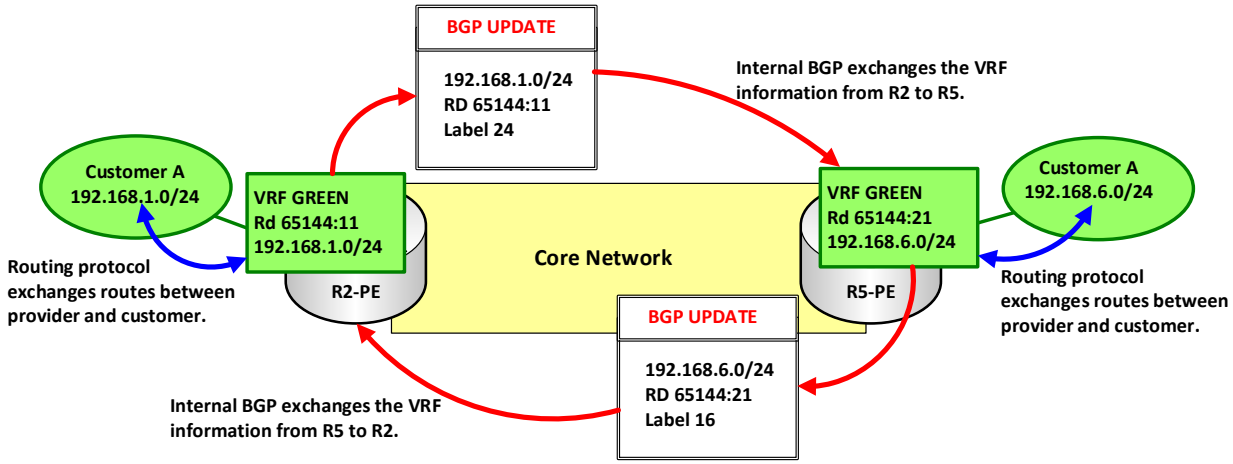


Figure 11.19

The PE routers populate their VRF tables with the information gained. Now, both routers know all the routing information necessary to allow traffic to travel across the core network. Fundamentally, the two ends of a virtual private network (VPN) have been tied together.

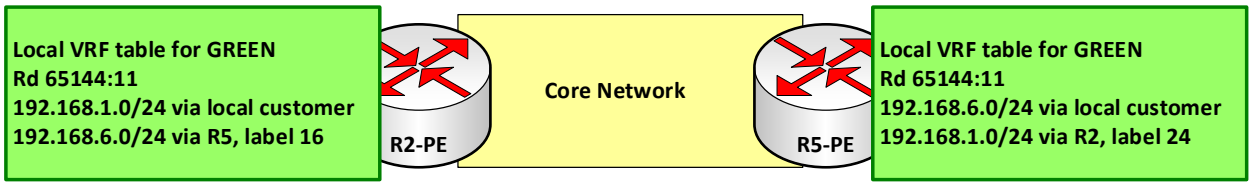


Figure 11.20

When the process is completed for all the VRFs, virtual private networks are completed. From the viewpoint of each customer, there is a long connection running from site to site.

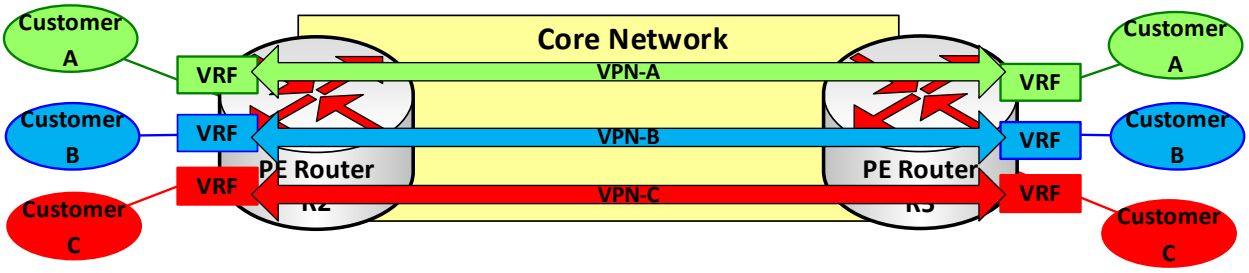


Figure 11.21

Now that all the pieces are completed, the flight of data from one side to another can be explained. In the next figure, a computer with the IP address 192.168.1.10 is sending a TCP segment with data to another computer with the IP address 192.168.6.20, located in site 2.

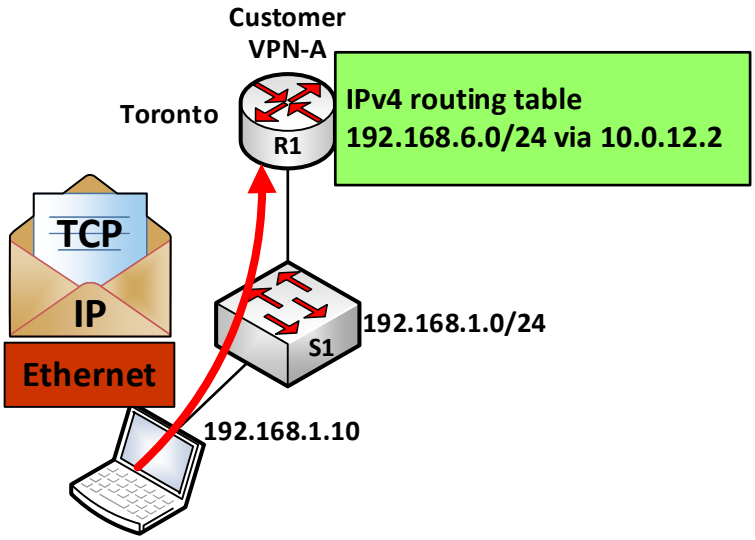


Figure 11.22

The Ethernet frame arrives to the default gateway R1 which proceeds to read the destination IPv4 address and then it looks that up in its IP routing table. It finds a match with a next hop of 10.0.12.2. Then R1 continues the process by delivering another Ethernet frame with the IP packet inside to the next router R2 which is a PE router. From here, the explanation concentrates in the IP packet without showing the encapsulation inner details.

The IP packet is received by the PE router R2. This router checks its VRF table and it finds a match for the destination IPv4 address. According to the same table, R2 needs to encapsulate the IP packet inside an MPLS packet with the label 19. It must deliver such MPLS packet to the router that advertises the destination address and that is router R5.

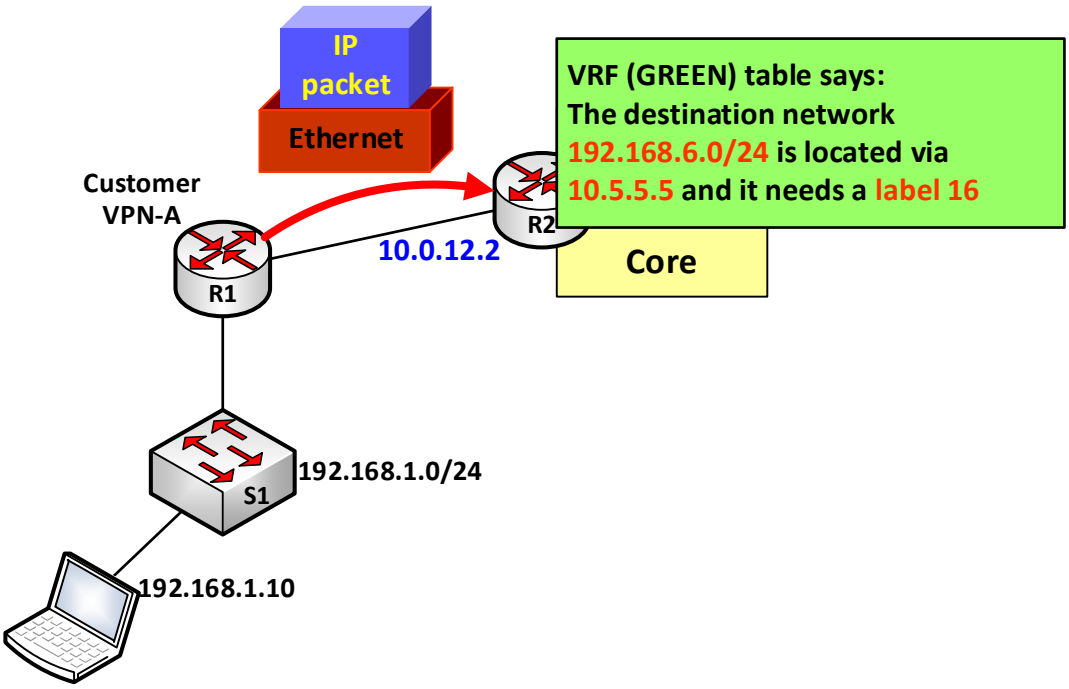


Figure 11.23

Router R2-PE places the IP packet inside the payload of an MPLS packet and it pushes the label 16 onto the header of the MPLS packet. The following diagram shows this action with the format of an MPLS header.

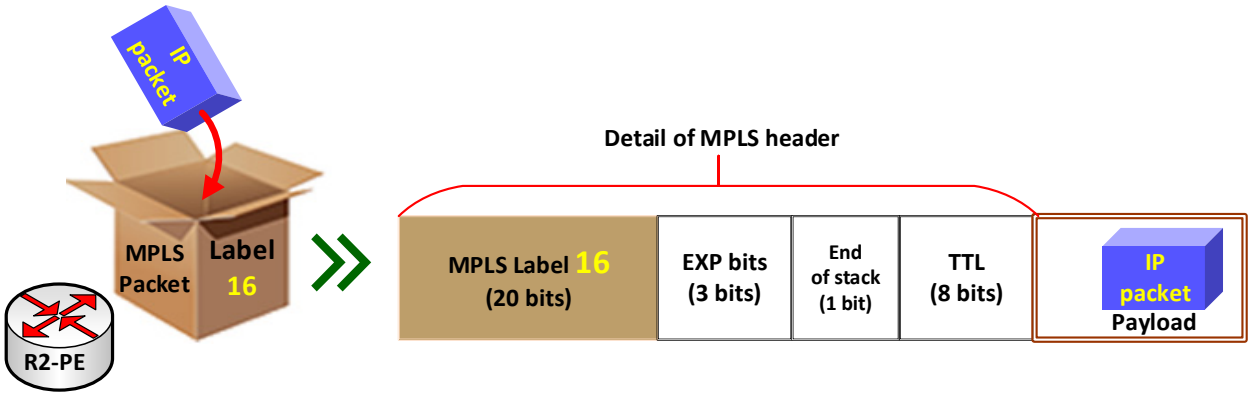


Figure 11.24

Now, R2-PE needs to send this MPLS packet to 10.5.5.5. The process of sending data from R2-PE to R5-PE across the core network was previously explained. So the same explanation is reused here. R2-PE checks its core MPLS table and it finds that to send anything to 10.5.5.5, it must deliver an MPLS packet with label 19 to the next hop router R3.

R2-PE#show mpls forwarding-table (edited)

Local-label	**Outgoing-label**	Prefix	Outgoing-interface	Next Hop (is R3)
22	19	10.5.5.5/32	f0/0	10.0.23.2

Then R2-PE takes the previous MPLS packet and places it inside another MPLS packet which is meant to go to R5-P. This is the critical part of understanding how this model works. There is a **stack** of two MPLS packets. The **outmost MPLS packet** is for crossing the core domain while the **inner MPLS packet** is to allow multiplexing of the different customers. The outmost MPLS packet is also called **top** MPLS or **core** MPLS packet, The inner MPLS packet is also called **bottom** MPLS or **VPN** MPLS packet.

The internal routers (P routers) switch the traffic using the top MPLS packets while the border PE routers sort out the traffic to the different customers by looking up the inner (VPN) labels. The P routers has no idea of what is outside the core because their routing knowledge is limited to the inside and that allows them to do a more efficient work of switching packets.

This is router R2-PE doing a double push to create the MPLS stack. The inner box with label 16 is only to be handled by the PE routers. The outer box is an MPLS packet with label 19 to be delivered to the next hop router R3-P. **Note**: of course, everything is always encapsulated by a layer 2 protocol like Ethernet.

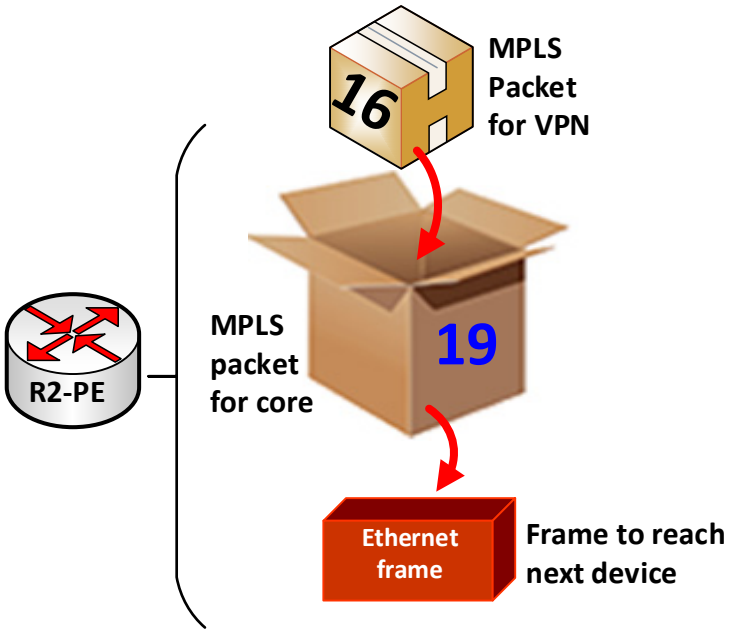


Figure 11.25

R3 understands that 19 means "delivery for 10.5.5.5". From here, the delivery decision is a manner of reading the label, swapping the label, until the **penultimate router** that **pops** the outer (also top or core) MPLS header.

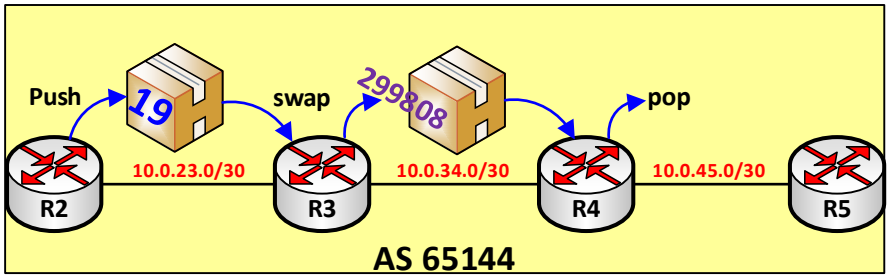


Figure 11.26

The pop action has to be examined in detail. R4 looks up its MPLS table and detects that the action is to pop the top MPLS packet leaving the inner MPLS packet exposed. In consequence, the destination R5-PE receives an MPLS packet with label 16.

felix@R4> show route forwarding-table (edited)

Destination	Next hop	Action	Net-if
299808 (S=0)	10.0.45.2	Pop	em0.0

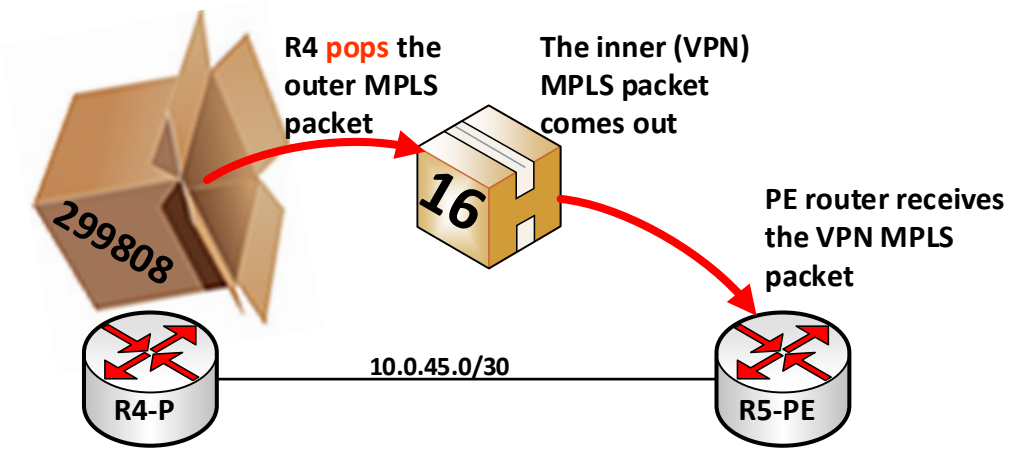


Figure 11.27

For R5, receiving an MPLS packet with label 16 means that the payload must go to VPN green. Consequently, R5 opens the last "box" and delivers the internal IPv4 packet onto the customer link and that is all that the client needs, its own IPv4 traffic.

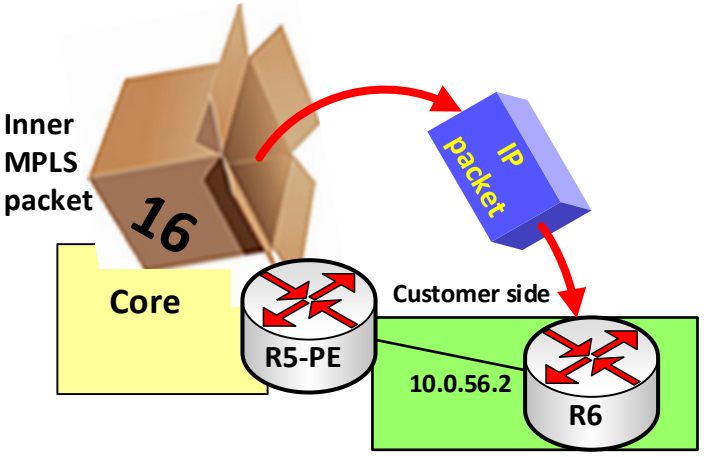


Figure 11.28

All the transactions are summarized in the following figure. (**Note**: the MPLS boxes are encapsulated by layer 2 Ethernet frames even though they are not shown in the diagram for the sake of simplicity.)

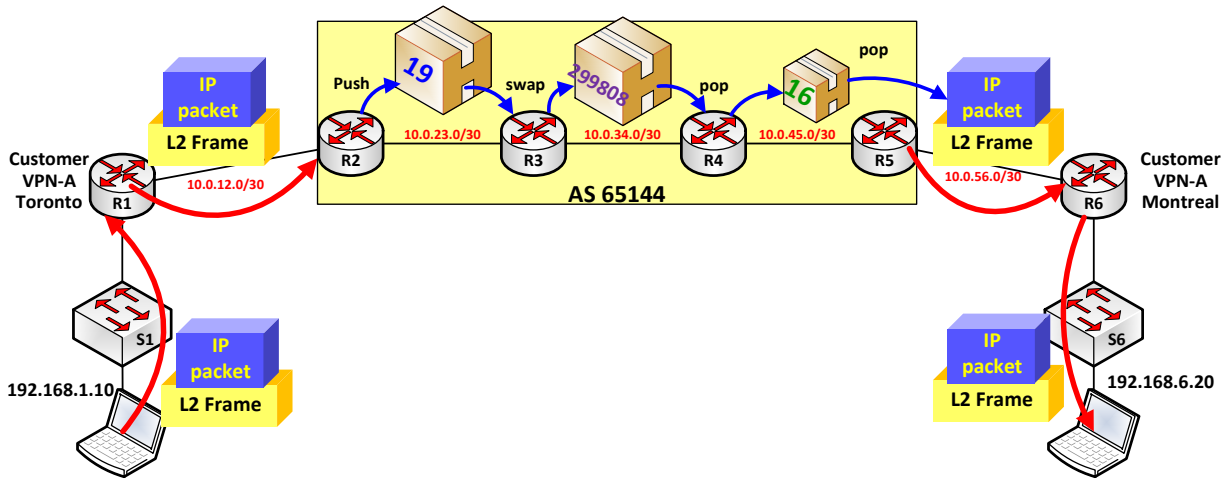


Figure 11.29

From the viewpoint of the customer, everything is just an IPv4 world. The client is unaware of what is happening inside the core. The important part is that the IP traffic flows transparently.

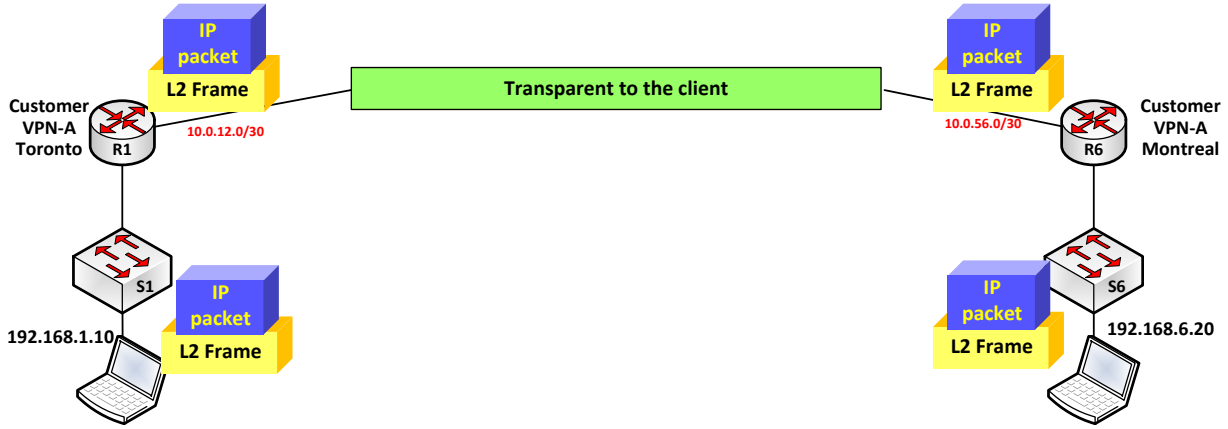


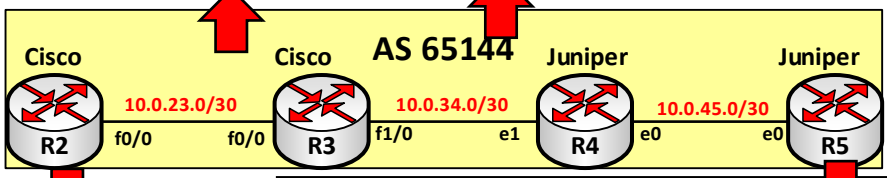
Figure 11.30

Configuration of an MPLS L3VPNv4 core

This is AS 65144 "skeleton" configuration regarding the interfaces and internal protocols.

```
hostname R3-P
!
interface lo0
ip address 10.3.3.3 255.255.255.255
!
interface f0/0
ip address 10.0.23.2 255.255.255.252
no shutdown
mpls ip
!
interface f0/1
ip address 10.0.34.1 255.255.255.252
no shutdown
mpls ip
!
router opsf 1
router-id 10.3.3.3
network 0.0.0.0 255.255.255.255 area 0
```

```
set system host-name R4-P
set interface lo0.0 family inet address 10.4.4.4/32
set interface em1.0 family inet address 10.0.34.2/30
set interface em1.0 family mpls
set interface em0.0 family inet address 10.0.45.1/30
set interface em0.0 family mpls

set protocols ospf area 0 interfaces all
set protocol ldp interfaces all
set protocol mpls interfaces all
```




```
hostname R2-P
interface lo0
ip address 10.2.2.2 255.255.255.255
!
interface f0/0
ip address 10.0.23.1 255.255.255.252
no shutdown
mpls ip
!
router opsf 1
router-id 10.2.2.2
network 10.2.2.2 0.0.0.0 area 0
network 10.0.23.0 0.0.0.3 area 0
```

```
set system host-name R5-PE
set interface lo0.0 family inet address 10.5.5.5/32
set interface em0.0 family inet address 10.0.45.2/30
set interface em0.0 family mpls
set interfaces em1 unit 0 family inet address 10.0.56.1/30

set protocols ospf area 0.0.0.0 interface em0.0
set protocols ospf area 0.0.0.0 interface lo0.0

set protocols ldp interface em0.0
set protocols ldp interface lo0.0

set protocol mpls interfaces all
```

Figure 11.30

The previous configuration files need to be commented in more detail. This figure compares a snippet taken from the Cisco router R2 and another from Juniper R5.

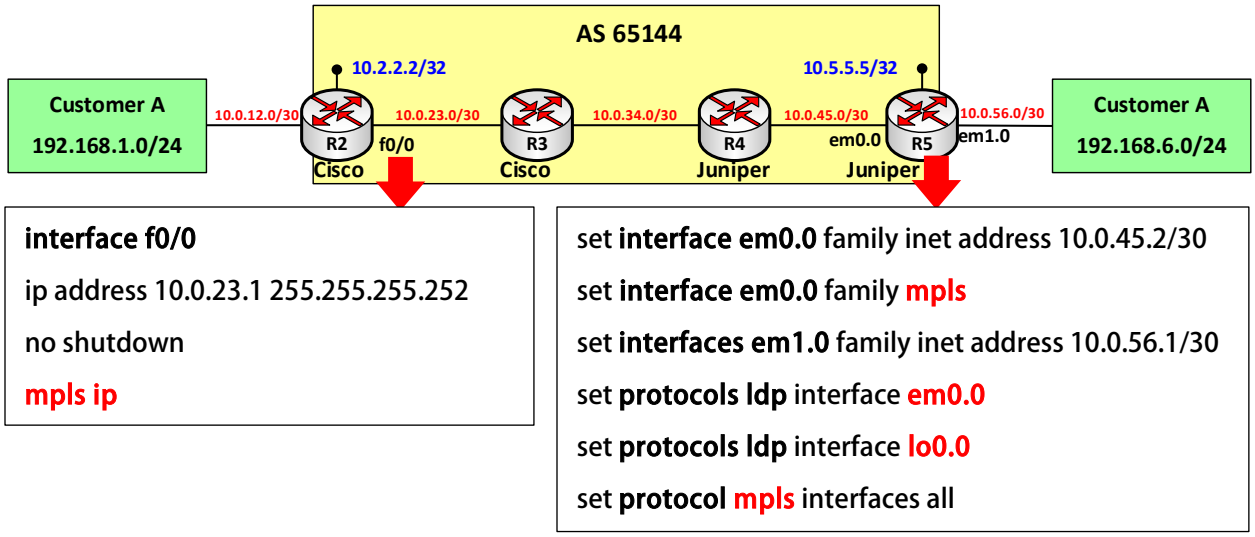


Figure 11.32

In the Cisco side, the command "mpls ip" achieves two actions. First, it activates the label distribution protocol LDP and second, it activates MPLS switching on that particular interface. Every physical interface inside the core must have this command. When LDP is activated, the router immediately starts sending LDP discovery messages to find potential LDP neighbours to exchange the labels. So switching MPLS packets and exchanging labels are two completely different processes even if they are both enabled with this simple Cisco command. Switching is part of a **forwarding plane** while exchanging information is part of a **control plane**.

In the Juniper side, the interfaces inside the core must have the "family mpls" command which activates MPLS switching. Also, the protocol MPLS must be activated in all the interfaces even the ones outside the core. This action enables the creation of labels including the VRF labels for the customer sides. In JUNOS, the protocol for label distribution must be configured independently from MPLS. If LDP is used then it must be enabled in every interface that participates in the label exchange process. The loopback interface must also be included because that is the router-id used to talk to other LDP routers.

The configuration of the Cisco router R2-PE is detailed starting with the configuration of the VRF named GREEN for Customer A. In Cisco IOS, a separated routing instance is defined under a name. A **Router Distinguisher (RD)** is assigned to differentiate this VRF from any other. Also, this RD is used to tie this side of the VPN with the other end of the VPN. This is accomplished by the commands **route-target export** and **route-target import**.

When a route-target is exported, it is sent inside a BGP UPDATE message as BGP community string. Hence, R2-PE is sending 65144:11 to the remote end R5-PE. On the other hand, R5-PE must send 65144:21 and R2-PE would accept that with the command route-target import.

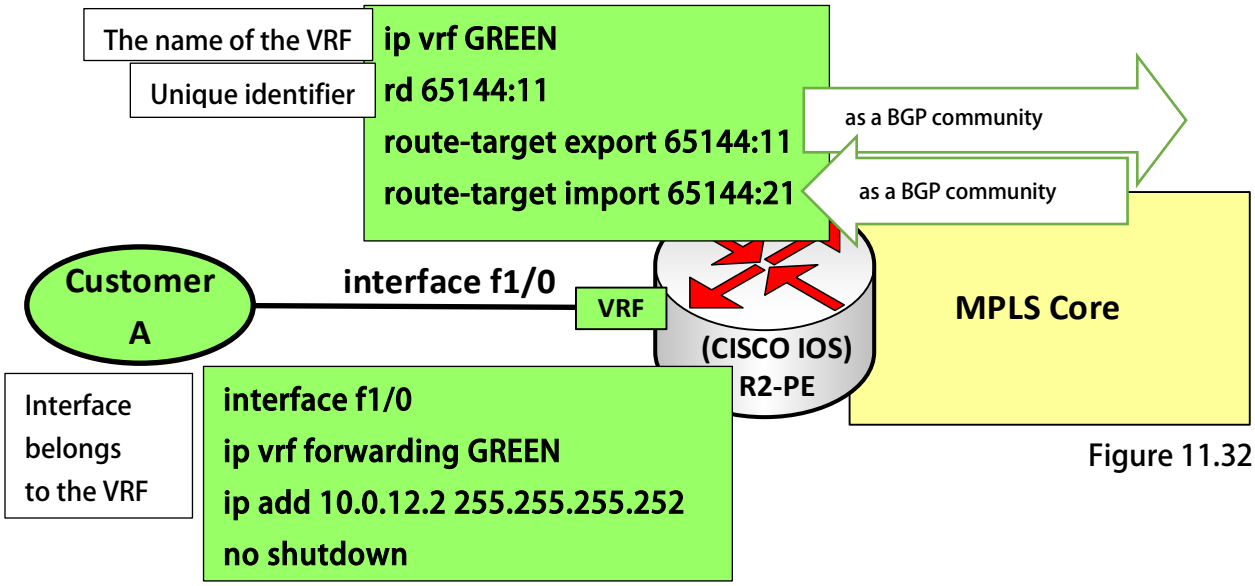


Figure 11.32

The links to the customer side must be made a participant of the VRF. By doing so, this network interface is no longer included in the default routing table and instead it is placed under the control of the VRF as these snippets prove.

R2-PE#show ip route 10.0.12.0 % Subnet not in table	R2-PE#show ip route vrf GREEN C 10.0.12.0/30 is directly connected, **f1/0**

The knowledge of the local VRF routing table must be transferred to the other PE router and vice versa. This is done with BGP and this is the configuration for Cisco IOS.

```
router bgp 65144
 no bgp default ipv4-unicast
 neighbor 10.5.5.5 remote-as 65144
 neighbor 10.5.5.5 update-source Loopback0
 !
 address-family vpnv4
  neighbor 10.5.5.5 activate
  neighbor 10.5.5.5 send-community both
 exit-address-family
 !
 address-family ipv4 vrf GREEN
  redistribute connected
  neighbor 10.0.12.1 remote-as 65001
  neighbor 10.0.12.1 activate
  neighbor 10.0.12.1 route-map NO_DC out
 exit-address-family
!
access-list 1 permit 10.0.12.0 0.0.0.3
!
route-map NO_DC deny 10
description "remove directly connected"
 match ip address 1
!
route-map NO_DC permit 20
description "allow routes from BGP"
!
```

The core's AS number is 65144.

The loopback addresses are used for iBGP peering.

BGP includes other address families.

VPNv4 is an address family that has its own routing profile.

The neighbour R5 is going to receive the route-target with a BGP community.

BGP must receive the routing information from the customer side. If the customer were using an IGP (any IGP, OSPF, IS-IS, RIP, EIGRP) then redistribution from the IGP into BGP would be defined here.

But in the current case, the customer speaks BGP also, so redistribution is not necessary. The easiest case to configure is when the customer also uses BGP instead of an IGP.

Customer A in Toronto side has the private AS number 65001.

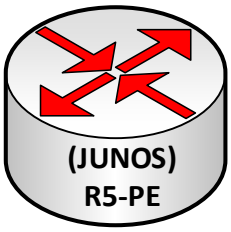
The network 10.0.12.0/30 is directly connected and it is being redistributed so the customer side in Montreal learns that too.

The directly connected route is being filtered with a route map so to avoid advertising it back to R1.

This concludes the configuration of R2-PE Cisco IOS.

R5-PE is a Juniper router. This is its configuration in detail.

```
set system host-name J5-PE

set routing-options router-id 10.5.5.5
set routing-options autonomous-system 65144
```




```
set protocols bgp group CORE type internal
set protocols bgp group CORE local-address 10.5.5.5
set protocols bgp group CORE neighbor 10.2.2.2 peer-as 65144
set protocols bgp group CORE family inet-vpn unicast
```

BGP spoken with neighbour R2 must be L3VPNv4 compatible

The routing instance for the VRF is defined now with the respective explanation below.

```
edit routing-instances GREEN
        set instance-type vrf
        set route-distinguisher 65144:12
        set vrf-export FROM-LOCAL-SITE
        set vrf-import FROM-REMOTE-SITE
        set vrf-table-label
        set interface em1.0
        set protocols bgp group GREEN-CUSTOMER type external
        set protocols bgp group GREEN-CUSTOMER neighbor 10.0.56.2 peer-as 65006
```

- The routing instance can be of different types (L2VPN, VPLS, etc). In this case, it is a routing-instance Virtual Routing Forwarding VRF.
- The route distinguisher for this VRF site is 65144:21. This uniquely identifies this customer GREEN in this Montreal site.
- The routing knowledge coming from the customer will be filtered, accepted, and marked with communities by the policy FROM-LOCAL-SITE to be exported to the customer GREEN in the Toronto site.
- On the other hand, the routing knowledge coming via iBGP from the remote PE router (R2) will be matched and accepted by the import policy FROM-REMOTE-SITE.
- In JUNOS, the command vrf-table-label creates one label for all the routes learned from the customer. In this case, two routes are injected, 10.0.56.0/30 and 192.168.6.0/24. (In contrast, Cisco IOS creates one label per IP prefix).

- The link that faces the customer must be added to the routing-instance. (In Cisco IOS, this action was executed under the interface configuration).
- Finally, the protocol being talked to the local customer. BGP is spoken with the customer router R6-CE in the autonomous system private number 65006.

The policy FROM-LOCAL-SITE is to receive the routes from the customer's local site which are going to be exported to the remote Toronto site.

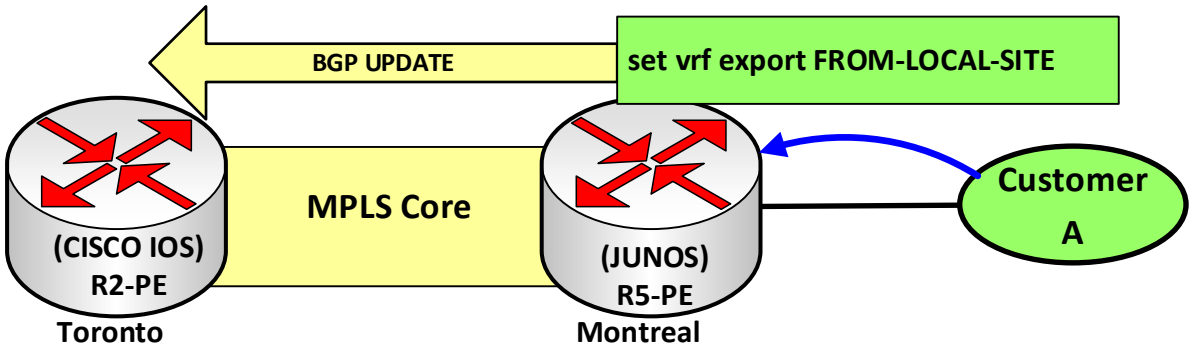


Figure 11.33

```
edit policy-options policy-statement FROM-LOCAL-SITE
        set term 1 from protocol bgp
        set term 1 then community add GREEN-LOCAL-COMMUNITY
        set term 1 then accept

        set term 2 from route-filter 10.0.56.0/30 exact
        set term 2 then community add GREEN -LOCAL-COMMUNITY
        set term 2 then accept

        set term 3 then reject
```

```
set policy-options community GREEN -LOCAL-COMMUNITY members target:65144:21
```

- First, the policy accepts any route advertised by the BGP protocol and then it attaches the community 65144:21 which is the value that R2-PE is expecting to import in the other site.

- The second statement matches the directly connected prefix 10.0.56.0/30 and it also sets the same community value. Nothing else is accepted.

The policy FROM-REMOTE-SITE is to receive the routes coming from the remote site, via iBGP that have the community 65144:11.

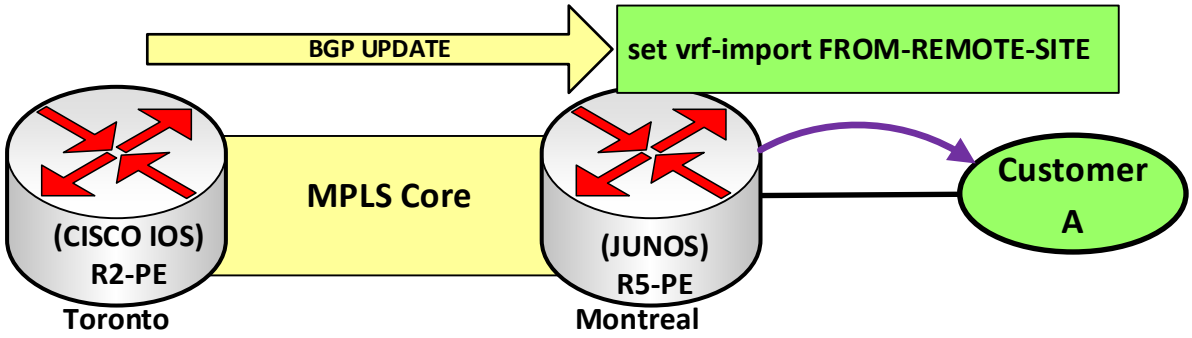


Figure 11.34

```
edit policy-options policy-statement FROM-REMOTE-SITE
        set term 1 from protocol bgp
        set term 1 from community GREEN -REMOTE-COMMUNITY
        set term 1 then accept

        set term 2 then reject
```

```
set policy-options community GREEN -REMOTE-COMMUNITY members target:65144:11
```

- Since R2-PE advertises these routes with the community 65144:11, it is just a matter of matching and accepting the protocol BGP and the correct community.

- The routes that pass the policy are installed in the local VRF table and from there, BGP carries them to the local customer site.

The customer sites are the only part that is needed to finish the configuration of this case.

```
hostname R1-CE
!
interface Loopback0
 ip address 10.1.1.1 255.255.255.255
!
interface FastEthernet0/0
 ip address 192.168.1.1 255.255.255.0
 no shutdown
!
interface FastEthernet1/0
 ip address 10.0.12.1 255.255.255.252
 no shutdown
!
router bgp 65001
 network 192.168.1.0
 neighbor 10.0.12.2 remote-as 144
```

```
set system host-name J6-CE
set interfaces em0 unit 0 family inet address 192.168.6.1/24
set interfaces em1 unit 0 family inet address 10.0.56.2/30
set interfaces lo0 unit 0 family inet address 10.6.6.6/32

set routing-options router-id 10.6.6.6
set routing-options autonomous-system 65006

set protocols bgp group AS65006 type external
set protocols bgp group AS65006 export TO_AS144
set protocols bgp group AS65006 neighbor 10.0.56.1 peer-as 144

set policy-options policy-statement TO_AS144 term 1 from route-filter 192.168.6.0/24 exact
set policy-options policy-statement TO_AS144 term 1 then accept
set policy-options policy-statement TO_AS144 term 2 then reject
```

Analysis of the BGP Conversation

This analysis begins from the BGP exchange between the customer R1-CE and R2-PE. After a standard BGP conversation, R2-PE places the routing knowledge gained in a special routing table which is the VRF for VPN GREEN. The following snippet shows that R2-PE knows the prefix 192.168.1.0/24 as advertised by R1's AS 65001 and the directly connected prefix 10.0.12.0/30 that was added via redistribution.

```
R2#show ip bgp vpnv4 all
Network          Next Hop     Metric     LocPrf     Path
Route Distinguisher: 144:11 (default for vrf GREEN)
 *> 10.0.12.0/30     0.0.0.0      0                     ?
*> 192.168.1.0/24   10.0.12.1     0                     65001 i
```

The next step is that R2-PE has to transfer this information to the other neighbour R5-PE. To do so, BGP is used as a signaling protocol to carry the routing information of the customers in the MPLS L3VPNv4 model. By talking iBGP, the PE routers delivers VPNv4 routes across the core network. This is possible because of the flexibility of the BGP UPDATE message that has been extended with two optional attributes: the Multiprotocol Reachability NLRI and the Extended Community.

The following snippet is a summary of a BGP UPDATE sent by router R2-PE to R5-PE (unnecessary details have been omitted). This BGP message carries all the default attributes as expected in any UPDATE plus two additional attributes.

BGP UPDATE (header)

Attribute ORIGIN is IGP
Attribute AS_PATH is 65001
Attribute LOCAL_PREF is 100
Attribute EXTENDED_COMMUNITIES
Attribute MP_REACH_NLRI

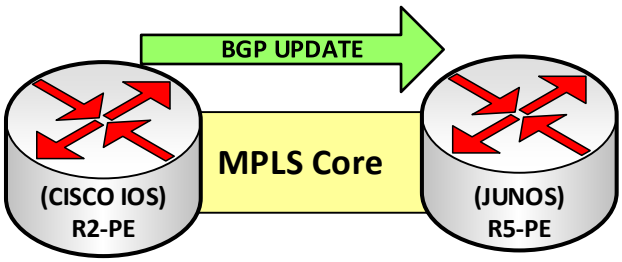


Figure 11.35

The **EXTENDED_COMMUNITIES** is BGP attribute code number 16. It is optional, transitive, and complete. This attribute has other applications beside L3VPNv4, in fact, it is quite open to applications. In the current case, the EXTENDED_COMMUNITIES is simply carrying the route-target that distinguishes the customer's VPN.

Attribute EXTENDED_COMMUNITIES
Route Target: 65144:11
AS specific: 65144 (2 bytes)
AN specific: 11 (4 bytes)

The **MP_REACH_NLRI** is the BGP attribute code number 14. It is optional, non-transitive and complete. This attribute carries all the information pertaining to the routes types. It describes several critical pieces of information like:

- The address family and whether it is unicast or multicast.
- The next-hop to reach the VPNv4 prefix. Here, R2 is the advertiser.
- The Network Layer Reachability Information that contains the IPv4 prefix, plus its route-distinguisher and the MPLS label for the inner (bottom or customer VPN) stack.

Attribute MP_REACH_NLRI
Address Family Identifier AFI: (1 = IPv4)
Subsequent Address Family Identifier SAFI: (128 = Labeled VPN Unicast)
Next Hop network address: Empty Label Stack RD = 0:0 IPv4 = 10.2.2.2
Network Layer Reachability Information NLRI:
Label Stack: **24** (bottom)
Route Distinguisher: **65144:11**
MP Reach NLRI prefix: **192.168.1.0/24**

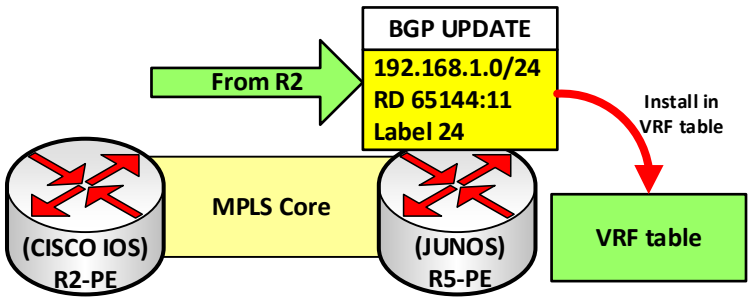


Figure 11.36

The second prefix 10.0.12.0/30 is advertised as well in a different BGP UPDATE.

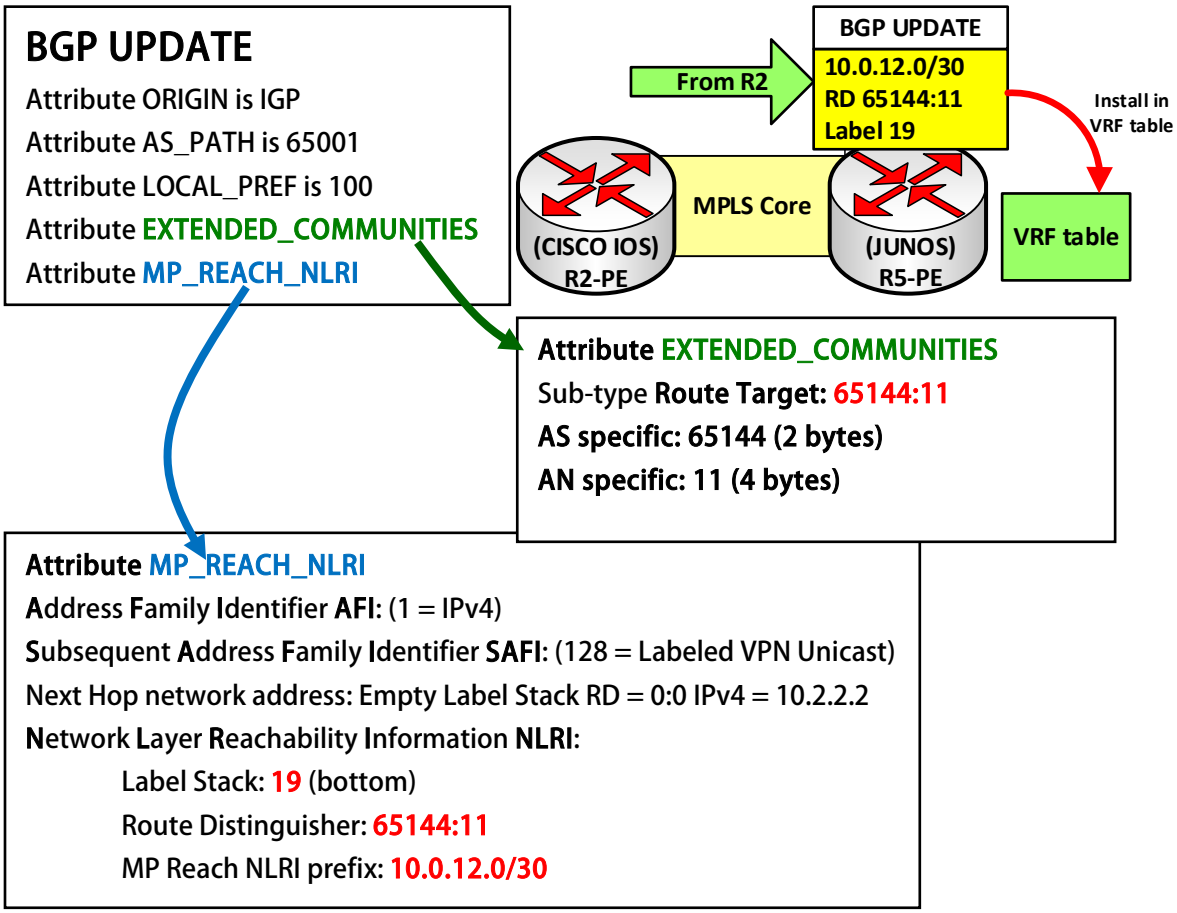


Figure 11.37

In the other side, R5-PE is a Juniper router with this acceptance policy that says (in pseudo-code): if the protocol is BGP, and the community is 65144:11, then accept and install that in the VRF routing instance table. Naturally, the two routes from R2 pass such filter.

In the meantime, R5 also has to advertise prefixes 192.168.6.0/24 and 10.0.56.0/30 to R2-PE. Two UPDATES are transmitted, one per each prefix, since some of the attributes are different.

BGP UPDATE

Attribute ORIGIN is IGP
Attribute AS_PATH is **empty**
Attribute LOCAL_PREF is 100
Attribute EXTENDED_COMMUNITIES
Sub-type **Route Target: 65144:21**
AS specific: 65144 (2 bytes)
AN specific: 21 (4 bytes)

Attribute MP_REACH_NLRI
Address Family Identifier AFI: (1 = IPv4)
Subsequent Address Family Identifier SAFI: (128 = Labeled VPN Unicast)
Next Hop network address: Empty Label Stack RD = 0:0 IPv4 = 10.5.5.5
Network Layer Reachability Information NLRI:
Label Stack: **16** (bottom)
Route Distinguisher: **65144:21**
MP Reach NLRI prefix: **10.0.56.0/30**

BGP UPDATE

Attribute ORIGIN is IGP
Attribute AS_PATH is **65006**
Attribute LOCAL_PREF is 100
Attribute EXTENDED_COMMUNITIES
Sub-type **Route Target: 65144:21**
AS specific: 65144 (2 bytes)
AN specific: 21 (4 bytes)

Attribute MP_REACH_NLRI
Address Family Identifier AFI: (1 = IPv4)
Subsequent Address Family Identifier SAFI: (128 = Labeled VPN Unicast)
Next Hop network address: Empty Label Stack RD = 0:0 IPv4 = 10.5.5.5
Network Layer Reachability Information NLRI:
Label Stack: **16** (bottom)
Route Distinguisher: **65144:21**
MP Reach NLRI prefix: **192.168.6.0/24**

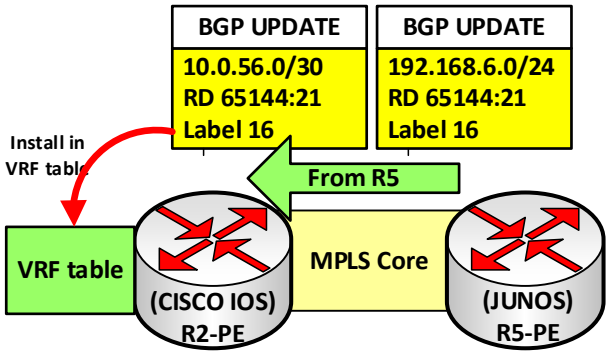


Figure 11.38

Notice that Cisco IOS assigns one label per prefix while Juniper JUNOS assigns the same label to all the prefixes from the same VRF.

Once that the BGP conversation is done, the two routers have a complete knowledge of the customer's routes.

```
R2#show ip bgp vpnv4 all
Route Distinguisher: 144:11 (default for vrf GREEN)
Network          Next Hop     Metric     LocPrf     Path
 *> 10.0.12.0/30  0.0.0.0      0                     ?
 *>i 10.0.56.0/30 10.5.5.5                100        i
 *> 192.168.1.0   10.0.12.1    0                     65001 i
 *>i 192.168.6.0  10.5.5.5                100        65006 i
Route Distinguisher: 144:12
Network          Next Hop     Metric     LocPrf     Path
 *>i 10.0.56.0/30 10.5.5.5                100        i
 *>i 192.168.6.0  10.5.5.5                100        65006 i
```

This is how R2-PE would make its decisions. If the received label is 19, then the payload (an IP packet) is sent to the directly connected interface. If the received label is 24, then the IP packet (no label outbound) is sent toward 10.0.12.1 as a next hop. Traffic outbound toward the remote location should be labeled 16 (inner MPLS packet) because that is what the router R5-PE wants.

```
R2#show ip bgp vpnv4 vrf GREEN labels
Route Distinguisher: 144:11 (GREEN)
Network      Next Hop     Incoming label     Outbound label
10.0.12.0/30 0.0.0.0      19                 no label (To local GREEN)
192.168.1.0  10.0.12.1    24                 no label (To local GREEN)
10.0.56.0/30 10.5.5.5     no label           16 (to remote site)
192.168.6.0  10.5.5.5     no label           16 (to remote site)
```

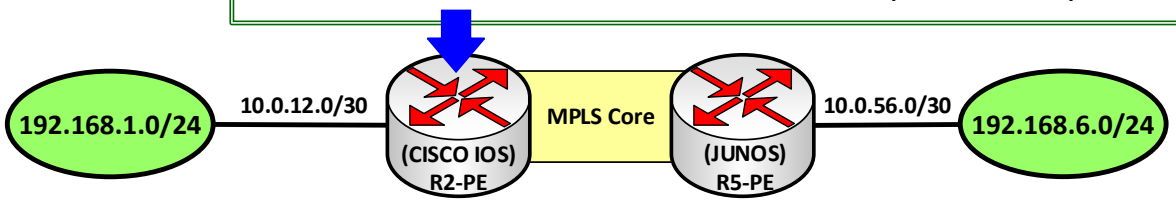


Figure 11.39

The customer routers only learn the following:

```
R1#show ip bgp
Network          Next Hop     Path
 *> 10.0.56.0/30  10.0.12.2    65144 i
 *> 192.168.1.0   0.0.0.0      i
 *> 192.168.6.0   10.0.12.2    65144 65006 i
```

```
felix@J6-CE> show route terse protocol bgp
A Destination      P Prf  Metric 1   Next hop   AS path
* 10.0.12.0/30     B 170  100        >10.0.56.1 65144 ?
* 192.168.1.0/24   B 170  100        >10.0.56.1 65144 65001 i
```

It only remains to test the IPv4 traffic from one end to the other by pinging and sniffing. This is an ICMP request (ping) from a computer connected in 192.168.1.0/24 to another computer connected in 192.168.6.0/24. The encapsulation is represented from the lowest layer (Ethernet or layer 2) to the upmost protocol layer.

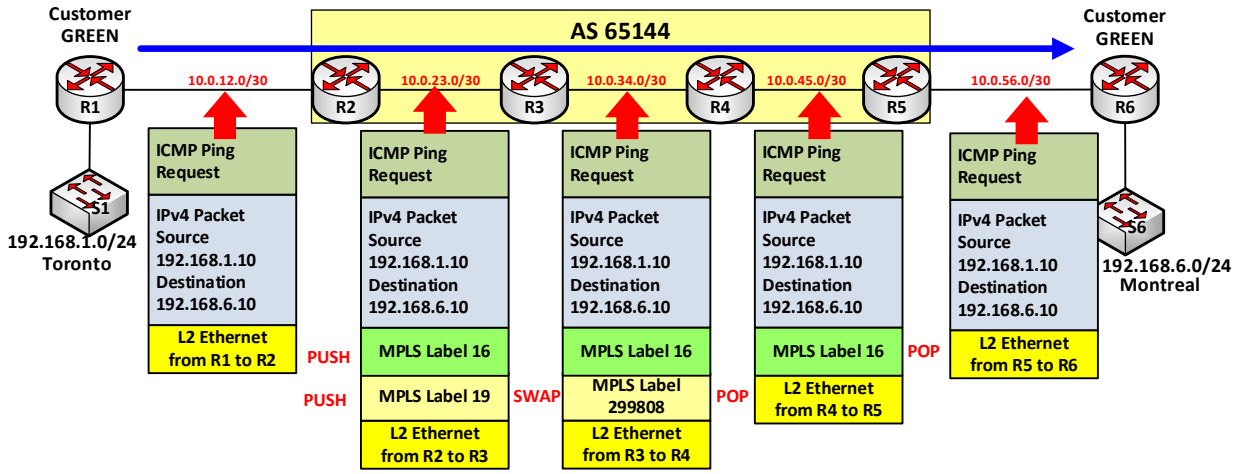


Figure 11.40

Finally, this is the ICMP reply from 192.168.6.10 to 192.168.1.10 to prove complete functionality of this model.

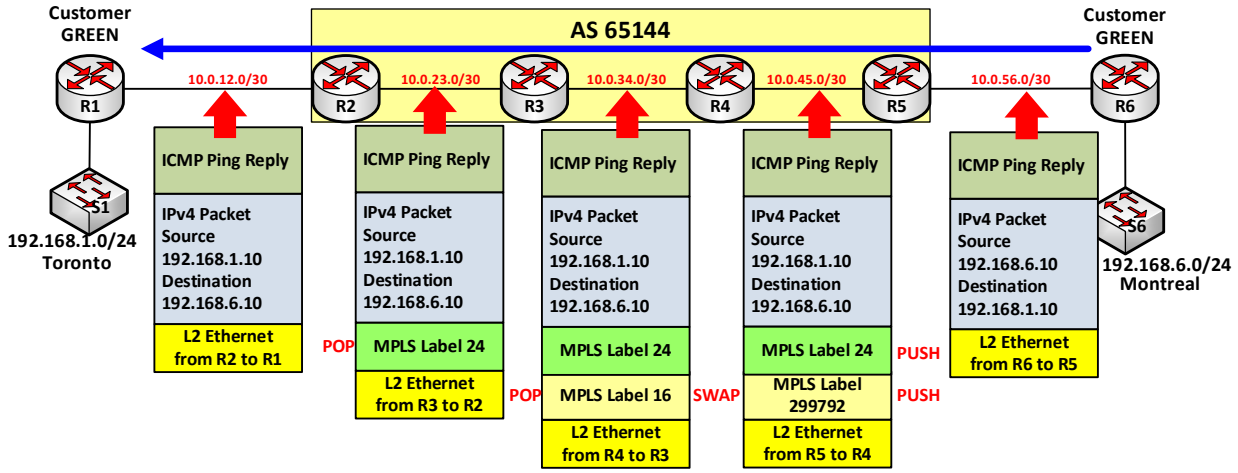


Figure 11.41

BGP signalling Layer 2 information

This section deals with a conceptual introduction to other BGP applications based on the same principle of using extended capabilities. It is quite remarkable that a routing protocol can be used to advertise other information beside IP routes. That is exactly what BGP can do and it all goes back to its designer's insight of making flexible messages that deal with objects with attributes. In fact, BGP has undergone an increased interest for applications in data centres and service provisioning models like the previously explained Layer 3 VPN MPLS. Beside that, BGP is also used to advertise:

- Layer 2 information in several applications such as:
  - Layer 2 VPN (L2-VPN Pseudo-wire)
  - Virtual Private LAN Service (VPLS)
  - Ethernet VPN (E-VPN)
- Security rules with BGP FlowSpec.

BGP signalling Layer 2 Virtual Private Networks

Let's begin with the definition of the problem. A company has a very simple network split in two locations. Such company only requires to interconnect the same layer 2 (LAN) segment.

Figure 11.42

There are many solutions to this problem but this section is only concerned with the ones that employ BGP to signal the presence of the customer. A service provider supplies a virtual wire end to end to interconnect the two LAN switches. Such "pseudo-wire" is signalled with BGP and that is the main topic of this explanation.

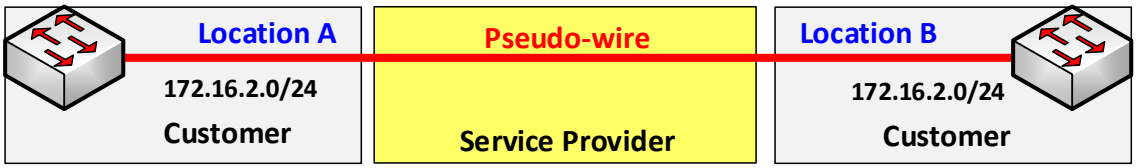


Figure 11.43

There are two solutions to the provisioning of L2 pseudo-wires. The oldest one is just point to point like in the previous diagram and it is called Layer 2 VPN Pseudowire. The second one is multipoint and multitenant and that is the model explained next.

Virtual Private LAN Service (VPLS)

Two customers, A and B, need to interconnect their sites using only layer 2 technology.

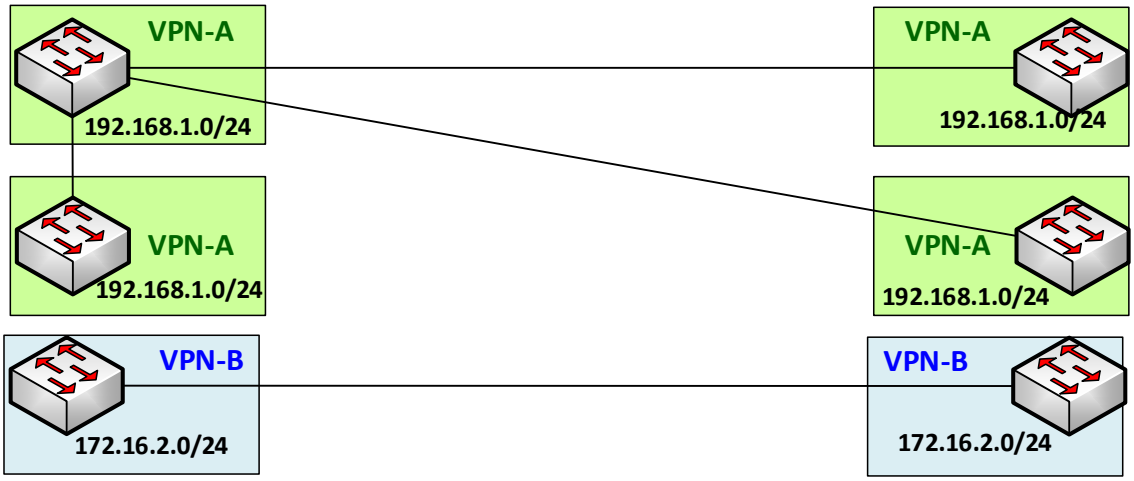


Figure 11.43

The solution is supplied by a service provider with an MPLS core (AS 65144).

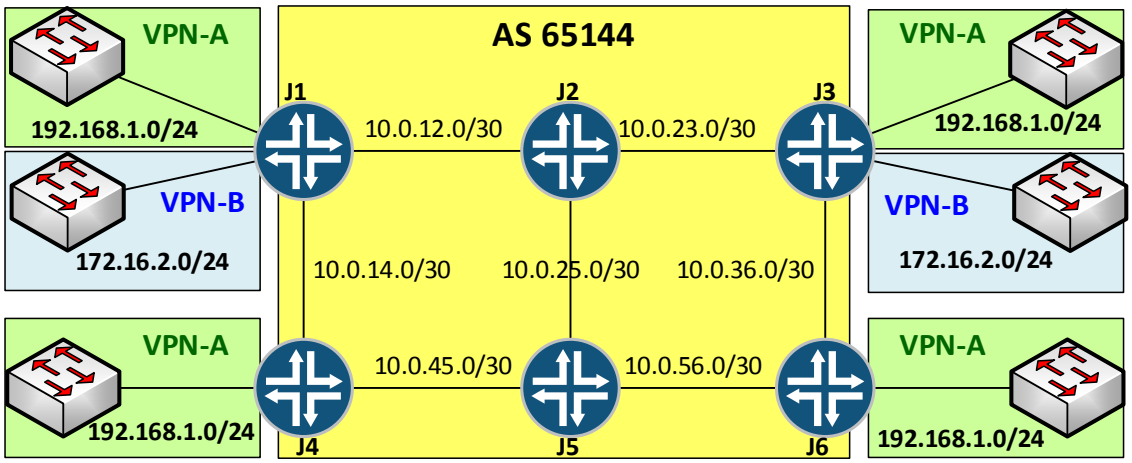


Figure 11.44

The service provider provisions tunnels that transparently mimic the layer 2 topology desired by the customer. These tunnels are overlaid on top of the service provider infrastructure. For the customer, it could not be more simple, it all looks like long cables interconnecting the sites.

The role of BGP in VPLS

The use of BGP as a signalling protocol for layer 2 virtual private networks was defined by K. Kompella and Y. Rekhter in RFC-4761 [3]. The key modification to the protocol was adding an Address Family number representing support for Layer 2 VPN. Then the three variations of the family (L2-point to point, VPLS, and EVPN) are described with Subsequent Address Families.

Border Gateway Protocol
Marker FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
Length: 59 bytes
TYPE: (number 1) OPEN Message

My BGP version is 4
My Autonomous System Number is 65144
My Hold Time is 90 seconds
My BGP identifier is **10.1.1.1 (this is J1)**
Optional Parameters Length = 30 bytes
Optional parameters:
Multiprotocol Extensions Capability
Address Family Identifier **AFI = (25) Layer 2 VPN**
Subsequent AFI **(SAFI) = (128) VPLS**

Edge router J1 starts the BGP conversation with an OPEN message. **L2 VPN**.

One relative of this family is **VPLS**.

The other routers of the service provider announce the same capability in their UPDATES.

Figure 11.45

Each customer-VPLS is assigned a domain wide unique identifier which is a route-target RT. The typical format for the RT is AS number: VPN number. For example, J1 has two VPLS attach to it, so it will advertise 65144:1 and 65144:2. The route target 65144:1 is the same for all the sites member of VPN-A. The routers deliver this RT with a BGP Extended Community.

Border Gateway Protocol
Marker FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
Length: 59 bytes
TYPE: (number 2) UPDATE Message

Path Attributes:
ORIGIN: IGP
AS_PATH: empty
LOCAL_PREF: 100

EXTENDED_COMMUNITIES:
Community **AS Route Target**: 65144:1

Community **Layer 2 Information**:
Transitive, Experimental
Subtype Layer 2 Information:
Encapsulation Type **VPLS (19)**

Multiprotocol_Reachability_ Information:
Address Family AFI: (25) **Layer-2 VPN**
Subsequent Address Family SAFI: (65) **VPLS**
Next hop network address: **10.1.1.1** (this is J1)
Network layer reachability information:
RD: 10.1.1.1:1
CE-ID: 1
Label-Block Offset: 1
Label-Block size: 8
Label Base: 100,000 (bottom)

Router J1 sends a copy of this UPDATE to all its peers, the edge routers J3, J4, and J6.

It advertises the Route Target 65144:1 which uniquely identifies the customer VPN-A. All other routers that have attached a site belonging to VPN-A recognize that they must adjoin the sites.

J1 describes a L2VPN VPLS indicating that J1 itself is a next hop to reach it.

J1 describes the NLRI in a very unique way. This part is critical for the functioning of the model and it requires a more detailed explanation next.

Figure 11.46

The VPLS model is meant to deal not with IP routes like a traditional routing protocol would do, but instead with layer 2 information and there comes a problem. Each site could potentially have a large number of MAC address and if all of them were to receive an MPLS label, then the routers would have very large tables that would consume a lot of resources. Instead, the designers of VPLS came up with a clever solution that is embedded in the fields of the NLRI.

Let's simplify the UPDATE to only show the relevant NLRI fields that J1 advertises to its peers:

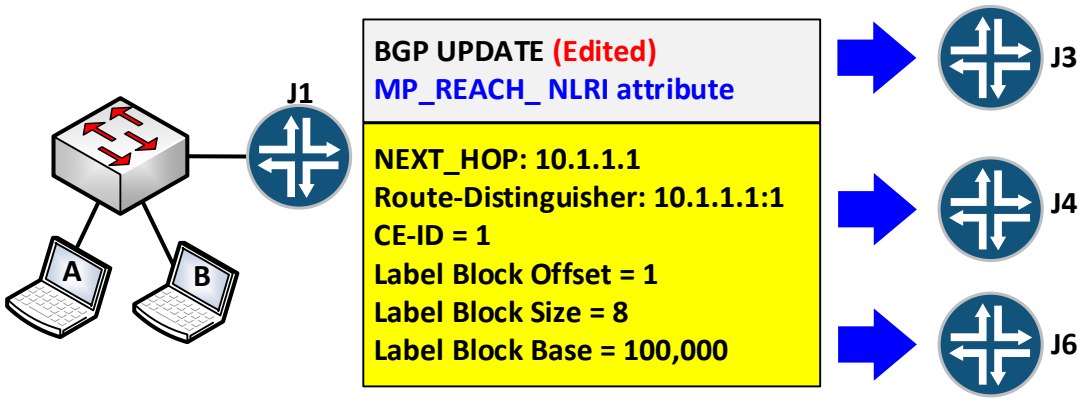


Figure 11.47

The UPDATE NLRI shows the:

- Loopback address of the advertising router.
- A route distinguisher assigned by the administrator during configuration. This RD has the number 1 to identify the VPN-A.
- A CE-ID or Customer Equipment Site Identification also assigned as per configuration. In this case, the site 1 of VPN-A. (in this example, site numbers are the same than router numbers to make it easier).
- Finally, a label block with a starting label number (100,000 in this example).

When the peers receive the update, they all perform the following calculation:

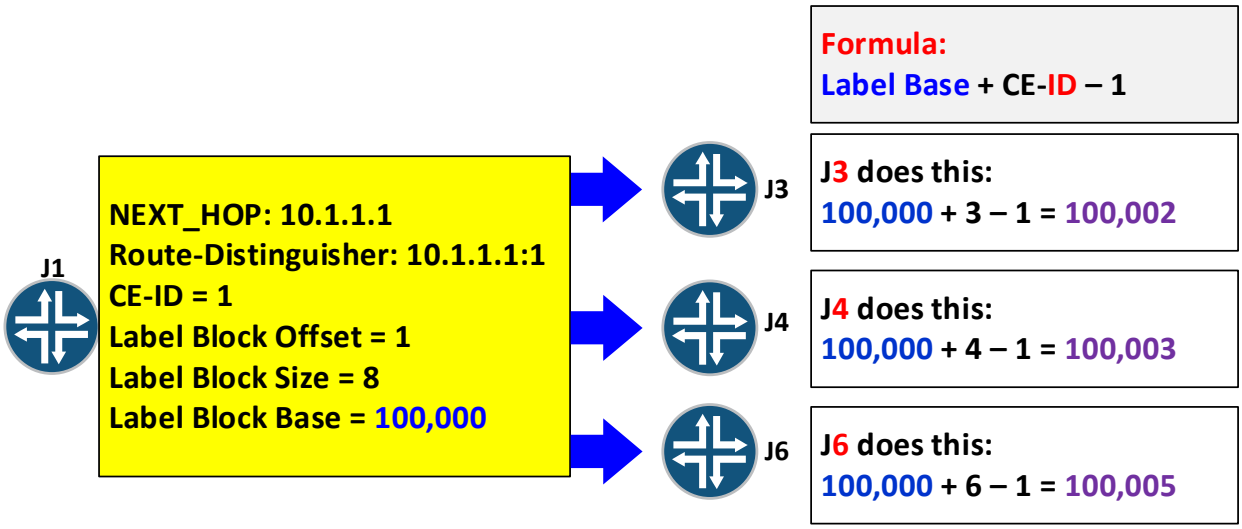


Figure 11.48

This label calculation system reduces the load on the routers. In effect, J1 does not need specific label advertisements per each MAC address located in the customer sites; instead, all the traffic that arrives to J1 with label 100,005 must necessarily be coming from J6. The label is implicitly pointing to the site and the customer.

To further elaborate, a flow of traffic must be observed and analyzed beginning with PC-A booting up. The first thing that PC-A will do is to send a DHCP request in broadcast mode. The VPLS edge router also has "learning switch" capability. Since the encapsulating L2 frame travels to all the ports in the LAN, J1 will listen and learn the MAC address of the sender of the frame.

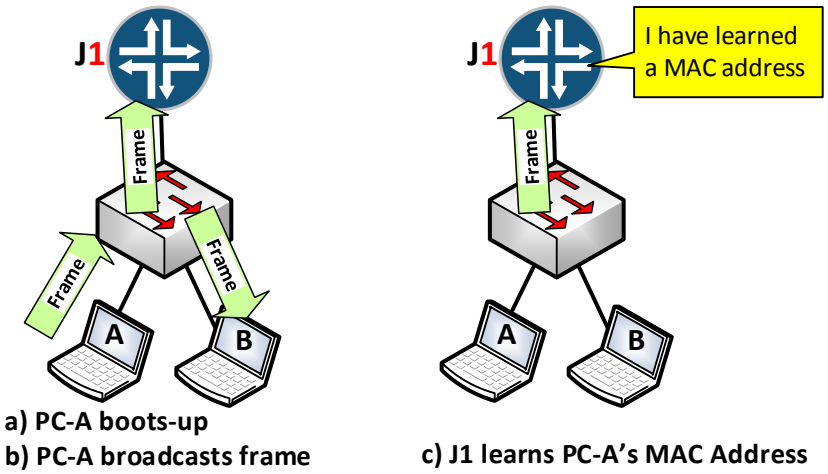


Figure 11. 49

By rule, J1 must forward the broadcast frame to all the sites that belong to the same VPLS. To simplify the explanation, only the transaction across J1 and J3 are shown next:

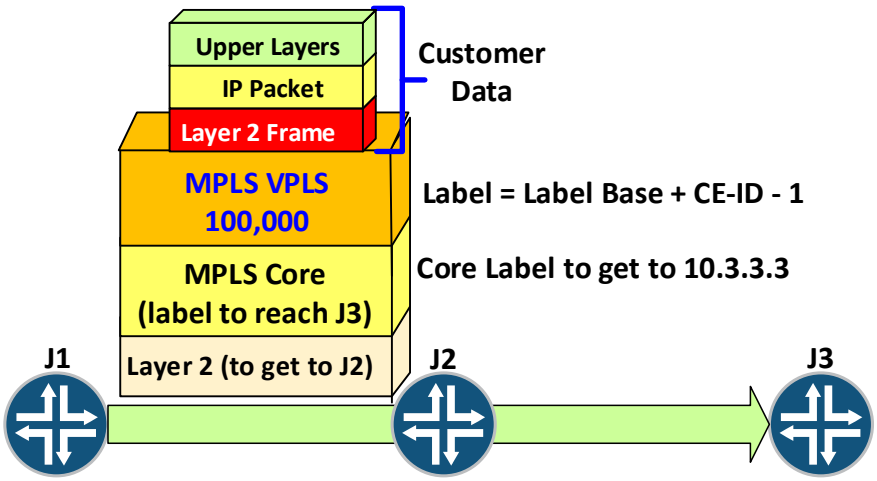


Figure 11.50

J2, being the penultimate router, pops the outer (core) MPLS header and it delivers the following frame to J3.

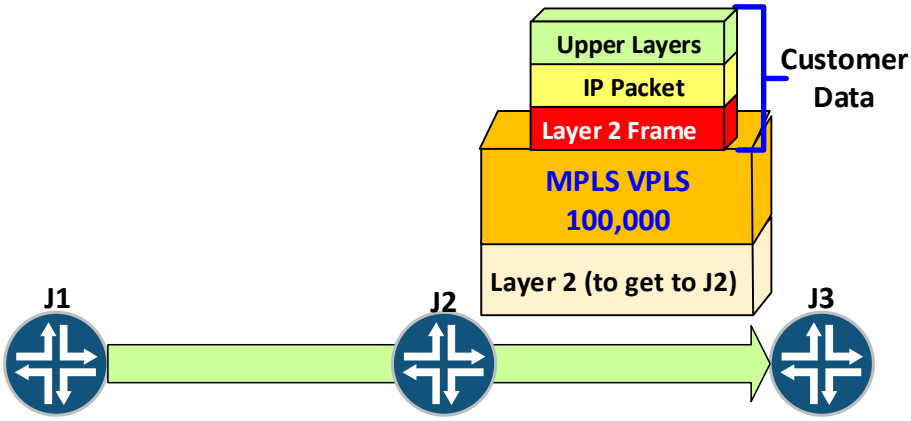


Figure 11.51

J3 receives the frame then it reads the MPLS header. The label 100,000 intrinsically carries the information of the VPLS identification and the router that is sourcing the data from the formula:

$$Label = Label\ Base + Site\ ID - 1$$

Basically, only J1 in site 1 for VPLS 1 could make a label 100,000. Subsequently, J3 forwards the encapsulated (VPN) Layer 2 frame into the corresponding interface.

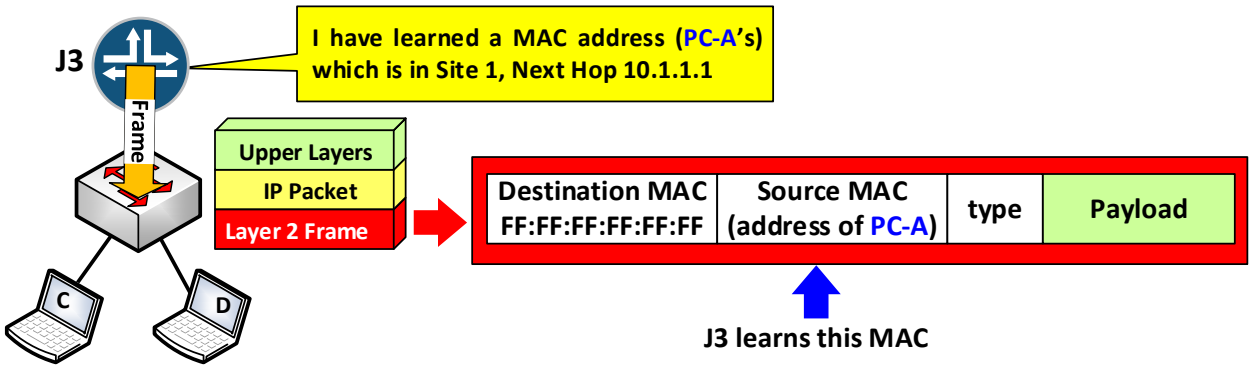


Figure 11.52

In the process, J3 also learns the MAC address of the source (PC-A) device and it ties all this knowledge together in a table. After this, J3 will be able to forward any traffic sourced from its local site CE-3 to the destination MAC address of PC-A.

For instance, host C in site 3, requires to send data to host A in site 1 by first time ever. At this moment, PC-C does not know the MAC address of PC-A so it issues an ARP request.

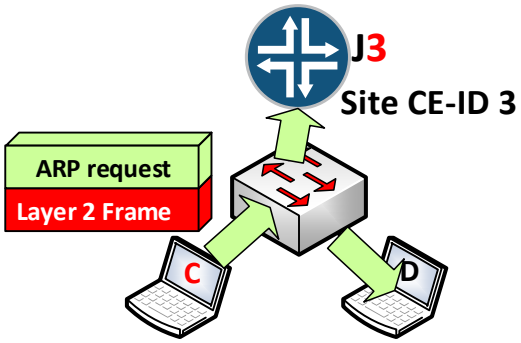


Figure 11.53

J3 proceeds to forward the broadcast ARP request to all the sites members of this VPLS. (even though J3 does know the MAC address of PC-A already but it can not correlate it to the question in the ARP request because it does not know PC-A's IPv4 address). For simplicity, only the transaction between J3 and J1 is shown next.

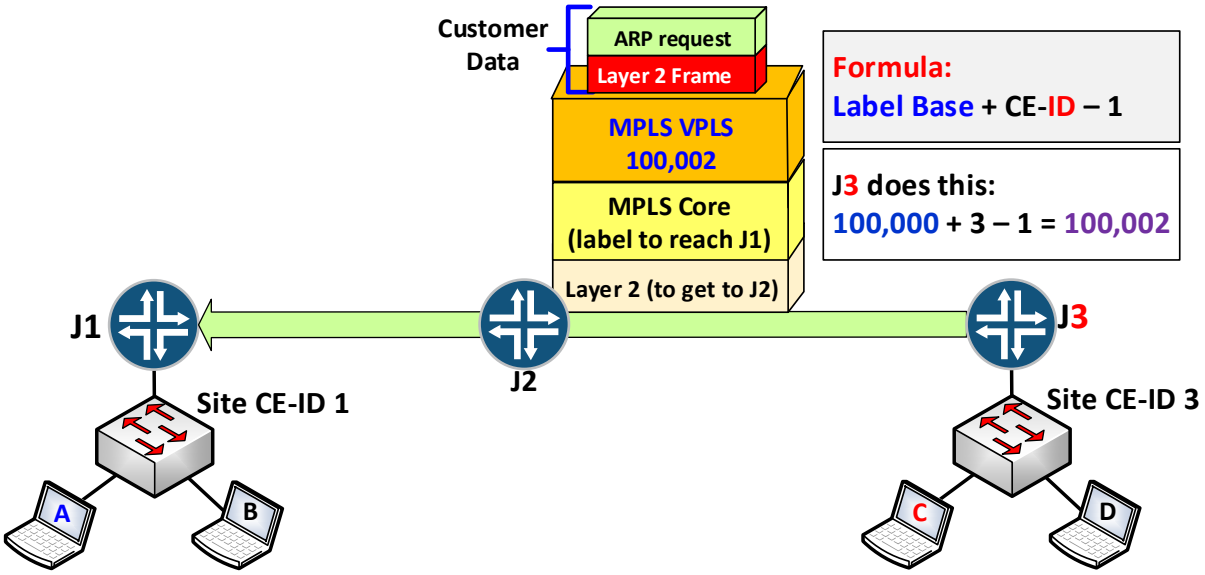


Figure 11.54

J2 pops the outer MPLS header and it exposes the bottom MPLS header with the label 100,002 to J1. This router recognizes that the source of the frame located in VPLS 1, site 3, next hop the loopback address of J3 (10.3.3.3).

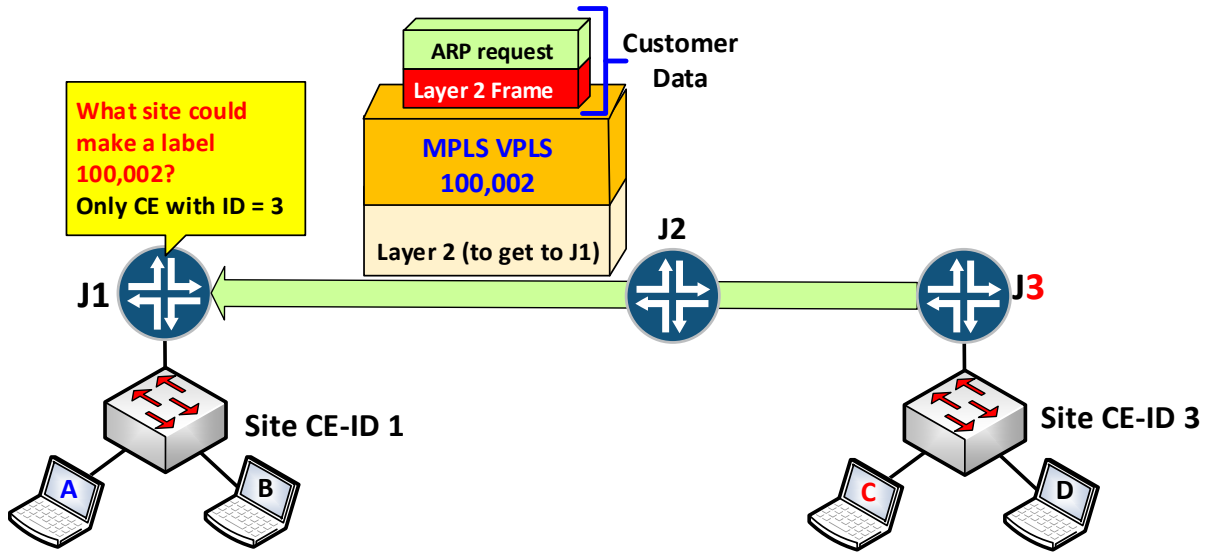


Figure 11.55

J3 forwards the customer frame toward the VPLS interface and in the process, it learns the MAC address of the sender PC-C.

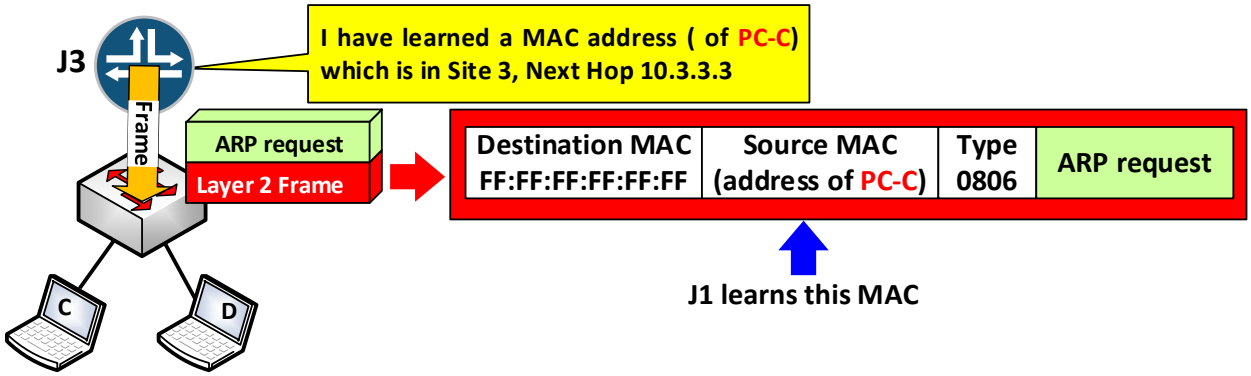


Figure 11.56

This is the way that the edge routers learn the locations of the MAC addresses of devices located in customer sites. The learning of the MAC addresses is similar to the way that LAN switches build their forwarding tables. In this model, BGP is just used to signal the location of the VPLS sites and the basic format of the label system without directly advertising each MAC address. This use of BGP [3] is called the "Kompella implementation" after one of its creators. The next section deals with another Layer 2 VPN model that actually advertises MAC addresses via BGP.

BGP signalling EVPN

Ethernet Virtual Private Networks (EVPN) is an extension of the idea that the proper encapsulation allow information to travel across networks even if those networks do not understand what is inside the packets. Fundamentally, EVPN allows the interconnection of a dispersed Layer 2 segment as if it were contained in one Ethernet based switch.

The problem that EVPN solves is the deployment of a layer 2 structure that is scattered across several devices and locations and the main application of EVPN is in data centres. For example, the following diagram shows three tenants (A red, B blue and C green) which are customers of a data centre.

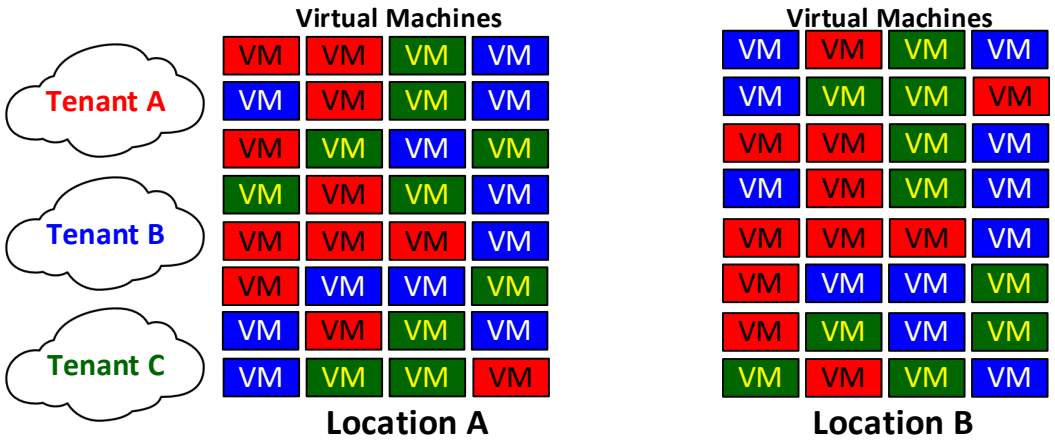


Figure 11.57

Each tenant has a large number of Virtual Machines (VM) which are dynamically created, modified, moved, or removed using an orchestration tool. It is common that one physical server machine contains dozens of virtual machines from different tenants. Furthermore, a typical data centre has several locations where the tenant resources are distributed for resilience reason and data is replicated among the locations for backups.

The tenant's virtual machines must be able to find each other regardless of their locations and the simple way to do that is to place them in the same Local Area Network LAN and let a basic protocol such as ARP to resolve their IP addresses to MAC addresses.

Now, to continue with this explanation, the previous diagram is converted to a more abstract view in the following figure. For each tenant, all of the virtual machines have been placed in its own layer 2 segment. Fundamentally, one Virtual Local Area Network (VLAN), per customer, that spans across physical servers, racks and even remote locations.

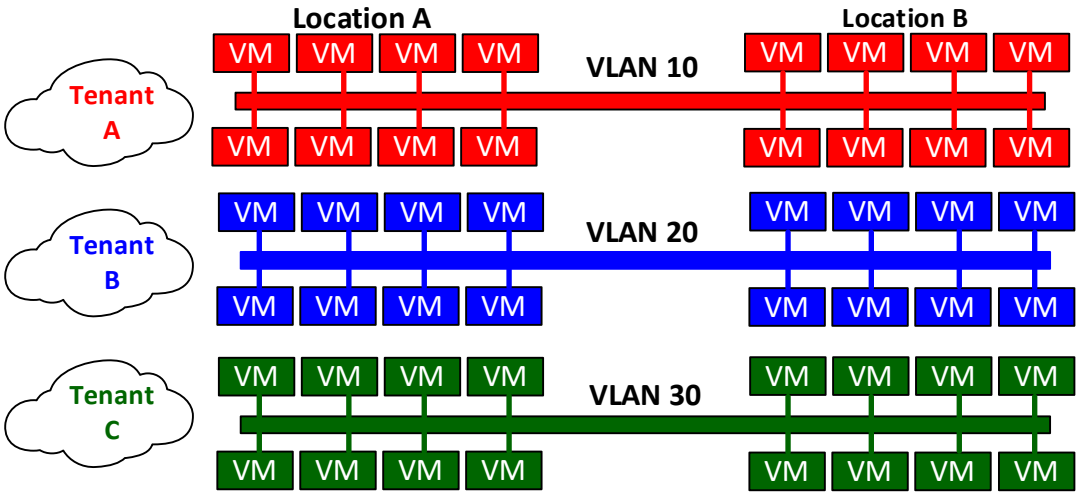


Figure 11.58

Clearly, something is missing in the picture, which is the way to interconnect the VLAN transparently regardless of the physical location. First, a **transporting network** must be in place between the two locations. That could be a simple IPv4 or IPv6 public network or it could be an MPLS based core. It does not matter as long as the edge routers can reach each other.

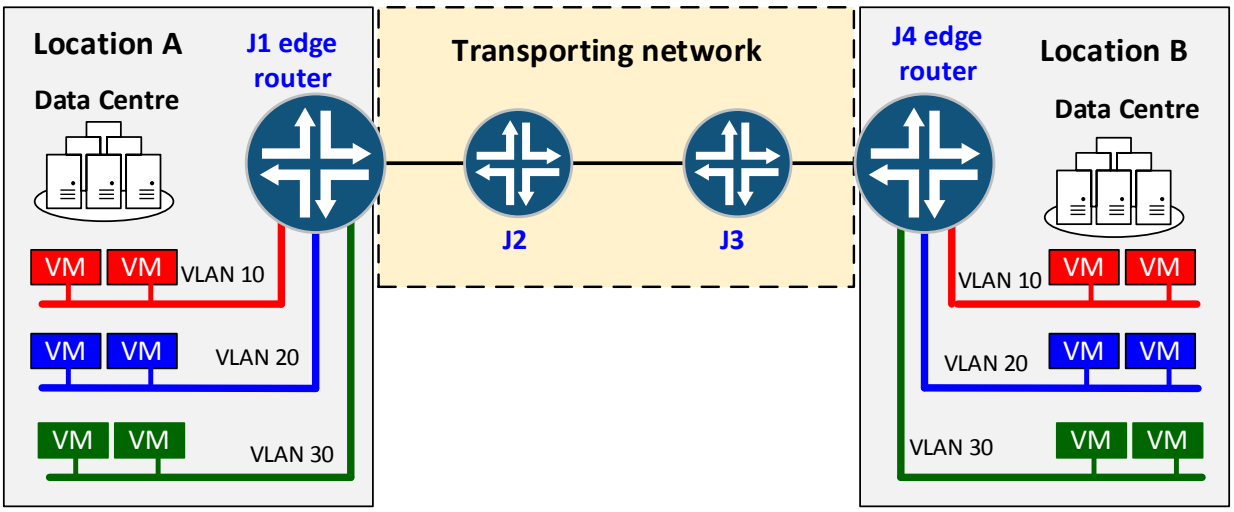


Figure 11.59

Second, there must a **mechanism to advertise** and propagate the information regarding the VLAN networks and host devices. There are several technologies that solve this problem, but one of them have proven to be more versatile and that is EVPN.

Ethernet Virtual Private Networks uses BGP as the mechanism to disseminate Local Area Network information. For example, edge routers J1 and J2 talk BGP to each other across the transport network to exchange their knowledge about the tenant's VLAN identification and the MAC addresses of the computers present in each VLAN.

Finally, the traffic will be encapsulated in a way that the core transport routers J2 and J3 can deliver without knowing anything about the final destination or the nature of the traffic. For example, VM-1 of tenant A (or red) sends data to VM-2 in the remote site B.

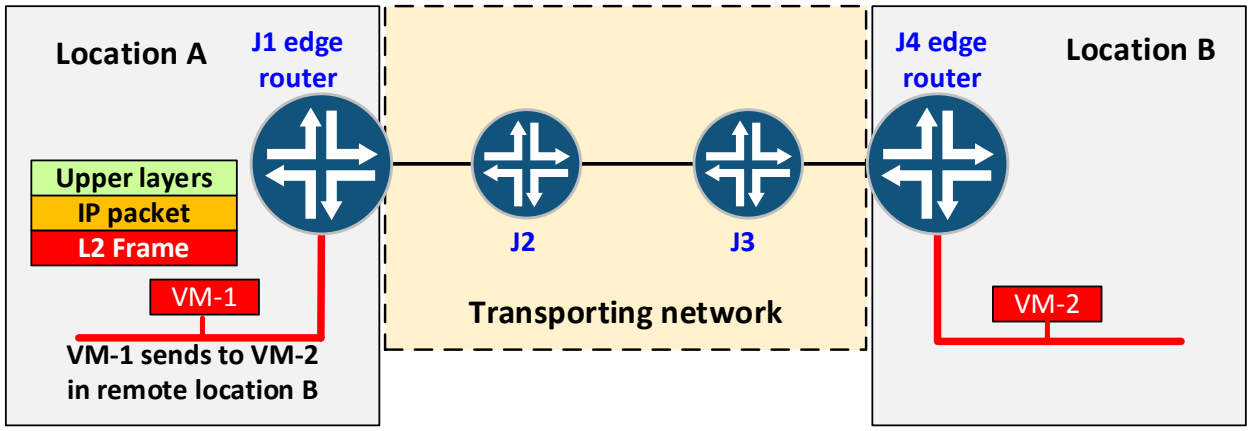


Figure 11.60

The edge router J1 receives the Ethernet frame and properly encapsulates it inside the payload of another protocol. When the transporting network is just IP then UDP is used to encapsulate the Layer 2 frame. The other option is to use an MPLS core in a manner similar to L3VPN MPLS.

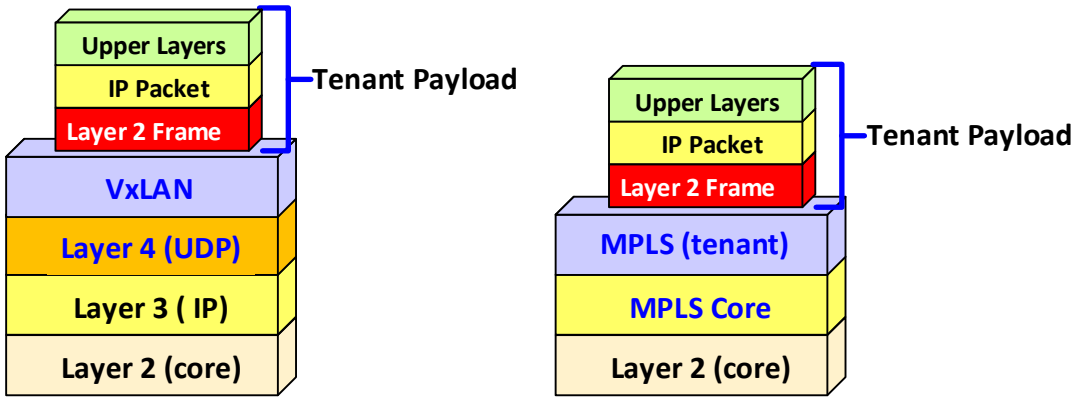


Figure 11.61

Regardless of the encapsulation model, the routers between the sites only need to know how to deliver the "encapsulating box" to the destination edge router (J4 in the diagram).

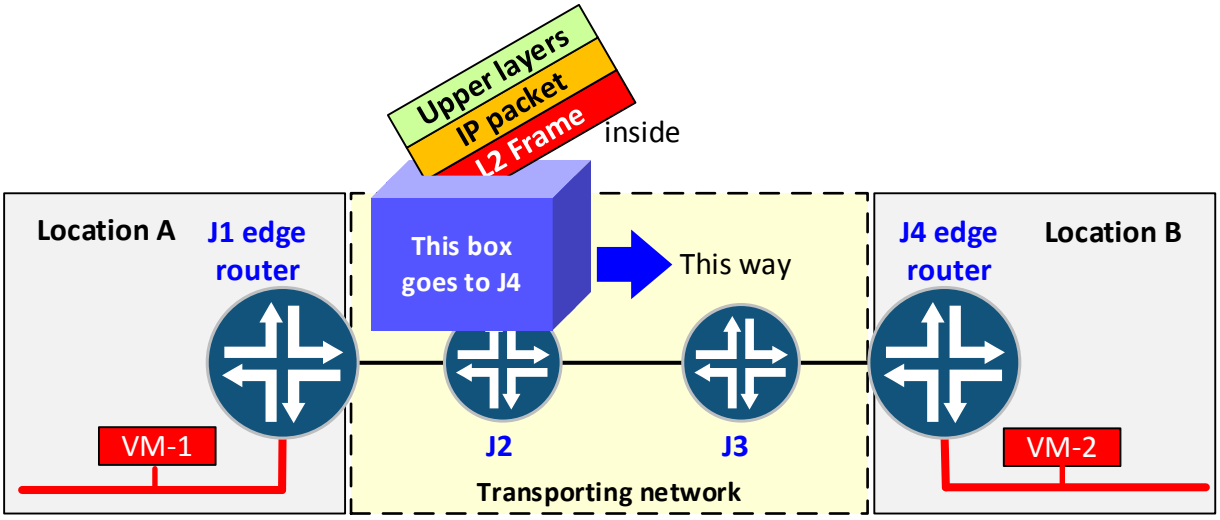


Figure 11.62

Edge router J4 receives the "box", decapsulates, reads the MPLS label, search its MPLS label table and then delivers the customer frame to the corresponding LAN segment.

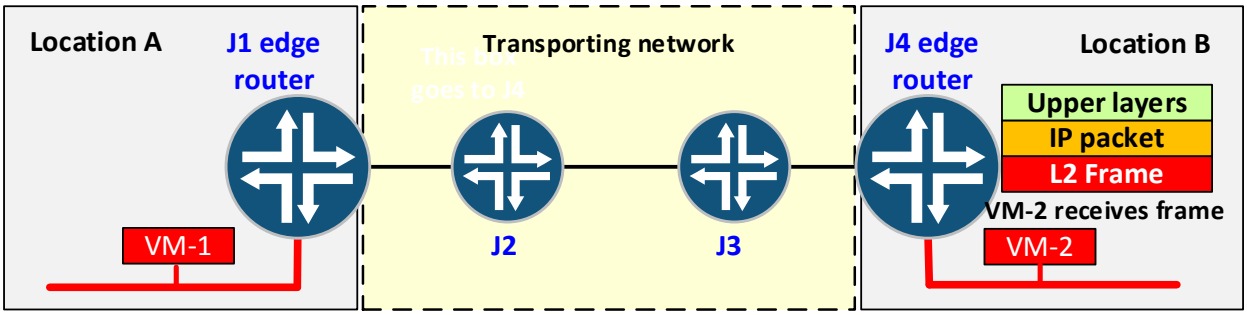


Figure 11.63

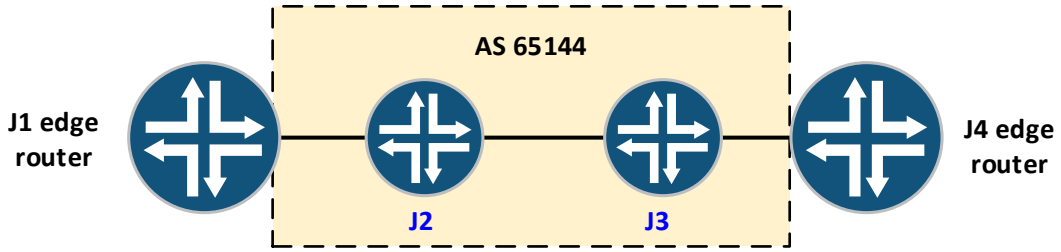
VM-2 receives the frame just as if it were connected to the same LAN switch than VM-1. The tenant is unaware of the external details. This is the network from the viewpoint of the tenant.

Figure 11.64

Once outlined the functioning principles of EVPN, the BGP conversation can be analyzed next.

BGP EVPN Conversation

The following sniffed snippet is the opening message that starts the conversation between edge routers J1 and J4. The other two routers J2 and J3 do not speak BGP at all.



Border Gateway Protocol
Marker FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
Length: 79 bytes
TYPE: (number 1) OPEN Message

My BGP version is 4
My Autonomous System Number is 65144
My Hold Time is 90 seconds
My BGP identifier is **10.1.1.1 (this is J1)**
Optional Parameters Length = 50 bytes
Optional parameters:
Multiprotocol Extensions Capability
Address Family Identifier AFI number 1 = IPv4
Subsequent AFI (SAFI) = 1 or Unicast

Multiprotocol Extensions Capability
Address Family Identifier **AFI number 1 = IPv4**
Subsequent AFI **(SAFI) = Labeled VPN Unicast (128)**

Multiprotocol Extensions Capability
Address Family Identifier **AFI = Layer 2 VPN (25)**
Subsequent AFI **(SAFI) = EVPN (70)**

A BGP OPEN message contains the basic introduction information such as the AS number, router-id, and hold time.

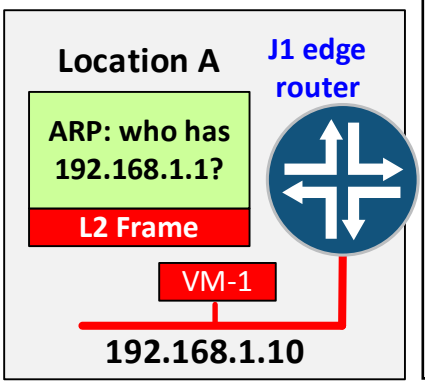
This OPEN message also describes the BGP session capabilities. The most elemental of which is IPv4 Unicast.

Additionally, this router has the capability to support IPv4 labeled VPN Unicast which means that it can implement L3 MPLS VPNs.

Finally, this router is capable of supporting another address family, the Layer 2 Virtual Private Networks, so it can MPLS **L2VPN**. One relative of this family is **Ethernet VPN**.

Figure 11. 65

The two edge routers mutually advertise their capabilities indicating that they can do EVPN. Once that they establish their BGP session, they proceed to exchange Layer 2 information but first they need to learn such information. This is the learning process graphically:



Assuming that virtual machine VM-1 of Tenant A (red) has just been booted up and it has obtained host address 192.168.1.10 and the default gateway IP address 192.168.1.1 from a DHCP server.

The next action for VM-1 is to try to find the default gateway's MAC address.

So, VM-1 just sends an **ARP request** message asking: **"who has 192.168.1.1? reply to me with your MAC address."**

Figure 11. 66

The ARP request is broadcasted and it reaches all the ports connected to the LAN including edge router J1's interface (J1 could or could not be the default gateway). In any case, J1 learns the MAC address of VM-1 and then it writes the entry in its own control plane table. In this way, edge routers are always learning MAC address just like any other computer in a LAN.

Next, router J4 proceeds to advertise the MAC address to its peer J4 using BGP. This is how the edge routers learn and propagate the knowledge about the VLANs and MAC addresses.

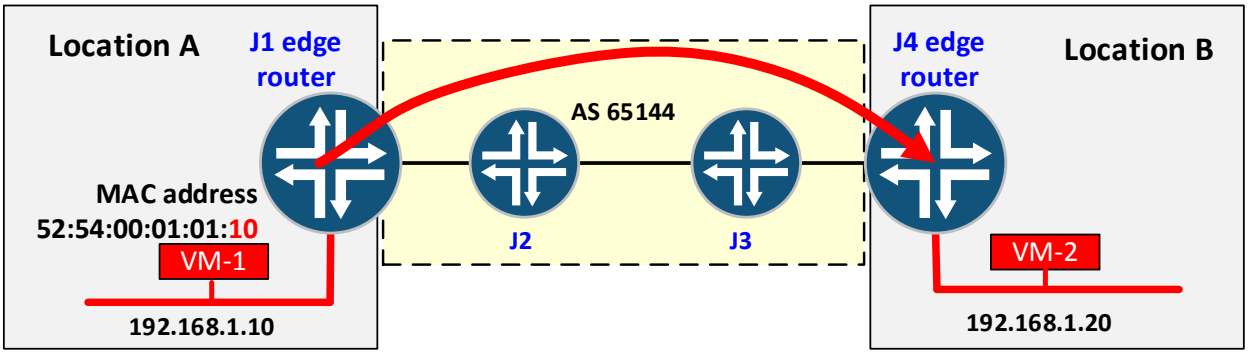


Figure 11.67

This is the BGP UPDATE (edited) message from J1 to J4:

Border Gateway Protocol
Marker FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
Length: 96 bytes
TYPE: (number 2) UPDATE Message

Withdrawn routes Length: 0
Path attributes:
ORIGIN IGP
AS_PATH: EMPTY
LOCAL PREF: 100
EXTENDED COMMUNITIES:
Community AS ROUTE TARGET: 65144: 144
Multi-Protocol Network Layer Reachability Information:
EVPN NLRI: MAC Advertisement Route
AFI: **MAC Advertisement Route** (2)
Route Distinguisher **RD: 65144:10**
Ethernet Segment Identifier ESI: 0
Ethernet Tag ID **(VLAN ID): 10**
MAC Address Length: 48 bits
MAC Address: 52:54:00:01:01:10 (MAC of VM-1)
IP Address: not included
MPLS **Label Stack: 300,010** (bottom)

Router J1 sends a BGP UPDATE to J4 advertising virtual machine VM-1 MAC address **52:54:00:01:01:10**

Tenant A has been assigned VLAN number 10. The field Tag advertises such VLAN number. This is how both edge routers interlace remote LANs.

The route distinguisher 65144:10 is used to disambiguate the tenant VLANs.

MPLS 300,010 is the label required to encapsulate traffic going to VM-1, VLAN 10 in location A.

Figure 11. 68

Both edge routers have learned VM-1's MAC address by now. Let's assume, for simplicity, that the same process have been completed for VM-2 MAC address. Now, an application in VM-2 needs to reach VM-1 but these machines have never talked before. So, VM-2 broadcast an ARP request asking for VM-1 MAC address since J4 knows the answer already, it acts as a proxy and replies back with the information.

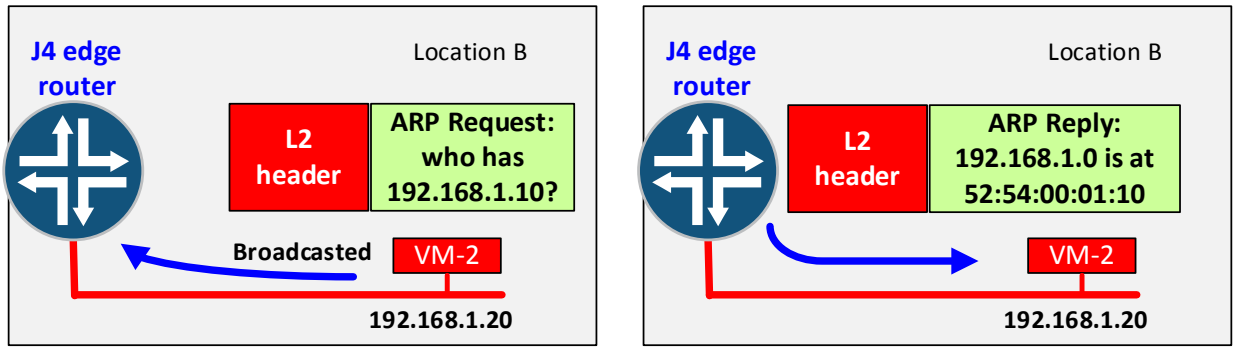


Figure 11.69

VM-2 places the destination MAC address of VM-2 in the header of an Ethernet Frame and encapsulates the rest of the protocol stack (IPv4, TCP and Application in this example), then it sends the frame to its connecting "learning bridge" switch. The frame is forwarded to J4.

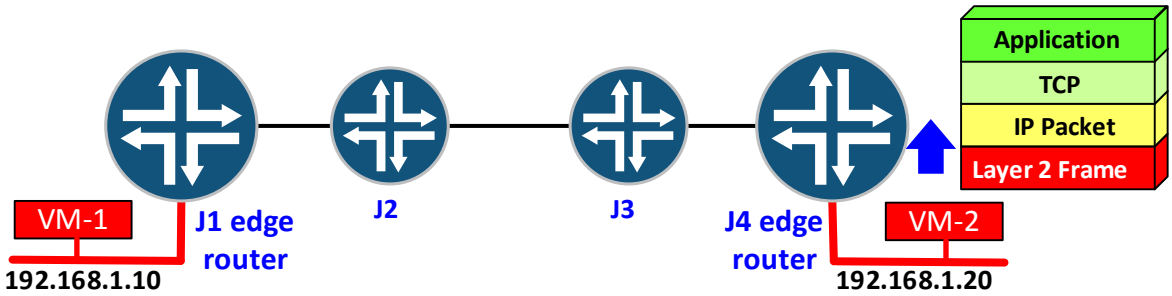


Figure 11.70

J4 modifies the frame by replacing the header with an IEEE 802.1q (trunking VLANs) header which contains the VLAN tag 10 and additional information. The source and destination MAC addresses of both VM-2 and VM-1 are also there.

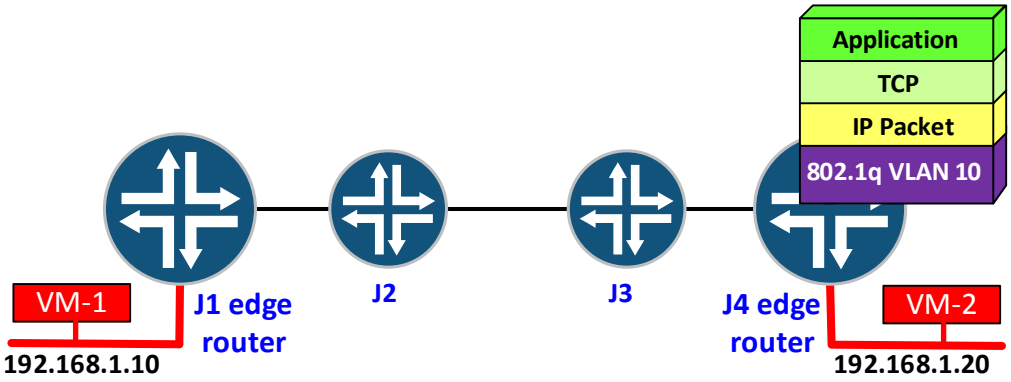


Figure 11.71

Furthermore, J4 adds the headers needed to cross the transport network, in this case an MPLS core. (For the IP core model, just replace the MPLS core header with UDP). J4 forwards the data to J3 with the destination label of J1.

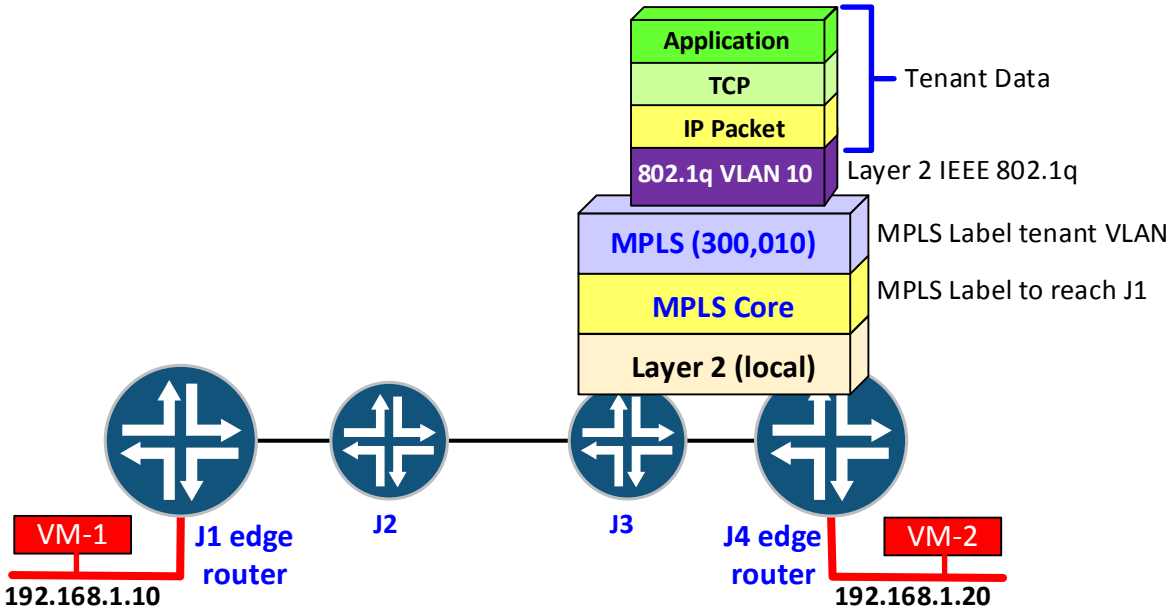


Figure 11.72

The core router J3 swap the MPLS core label forwarding the data to J2 which in turn, pops the MPLS core header and it delivers to J1 the following:

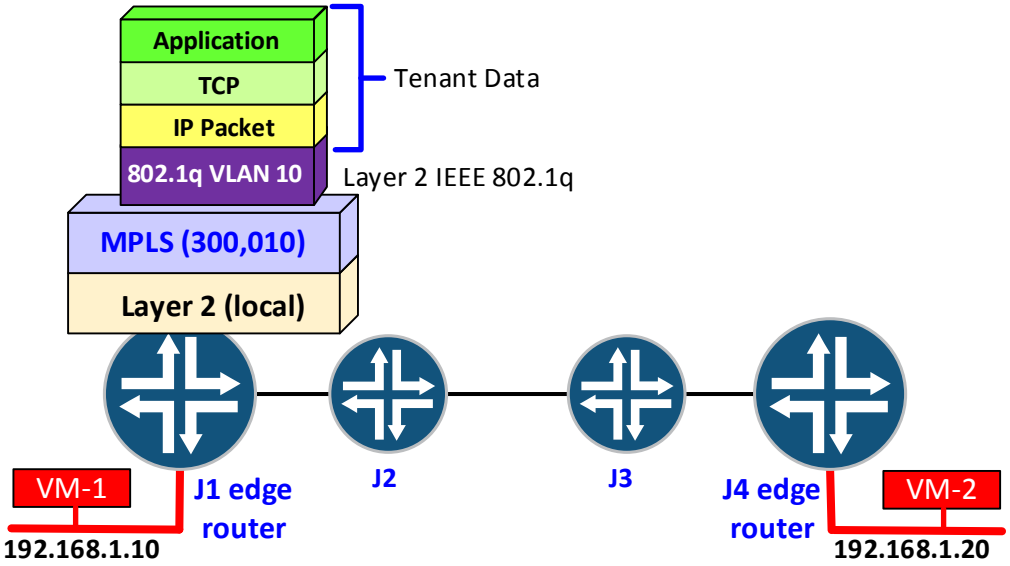


Figure 11.73

J1 reads the MPLS label and it proceeds to search its label forwarding database. The final label is meant to distinguish among the different tenants or VLANs. J1 finds out than label 300,010 is for EVPN Tenant A. The diagram does not show the physical detail of what happens next. The IEEE 802.1q frame is forwarded to a LAN switch (not shown). This LAN switch will change the trunked frame to an access type (IEEE 802.3) frame that the VM host can understand.

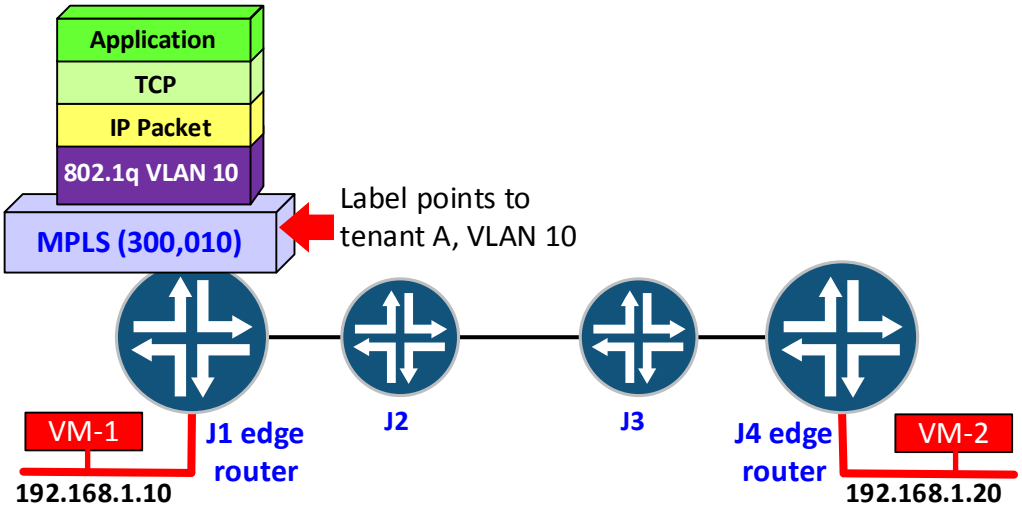


Figure 11.74

At last, VM-1 receives an IEEE 802.3 frame sourced by the MAC address of VM-2 with the destination MAC address of VM-1.

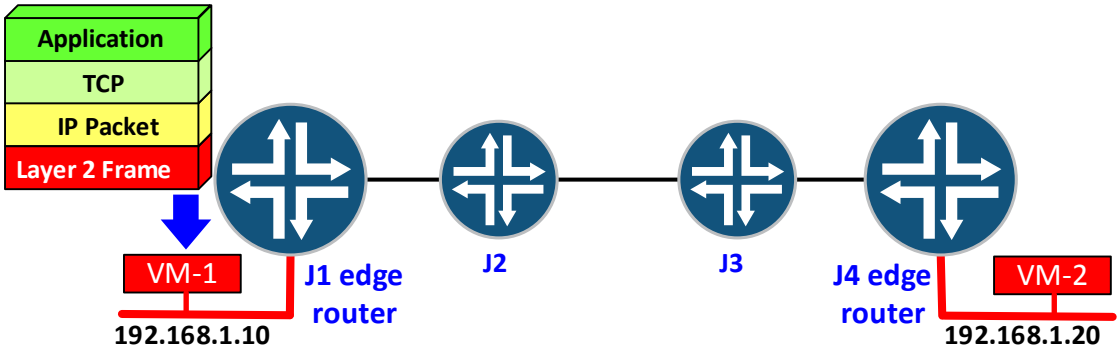


Figure 11.75

This concludes the overview of EVPN, a network model that has found wide acceptance in data centres. BGP sits at the core of this model using its control plane to advertise information about VLANs and MAC addresses. Something unsuspected for a routing protocol but because of BGP intrinsic design, it can be adapted to new applications by adding capabilities and attributes.

BGP signalling Security Rules with FlowSpec

The adaptability of BGP as a protocol is once more proven with the specification [5] of a way to propagate traffic flow control rules such as firewall rules, for example. Flow Specification or FlowSpec specifies [5] the use of the Network Layer Reachability Information NRLI to deliver rules that might block or allow specific patterns of traffic. Essentially, FlowSpec is a way to distribute traffic filters via BPG. A typical application is introduced in the following diagram.

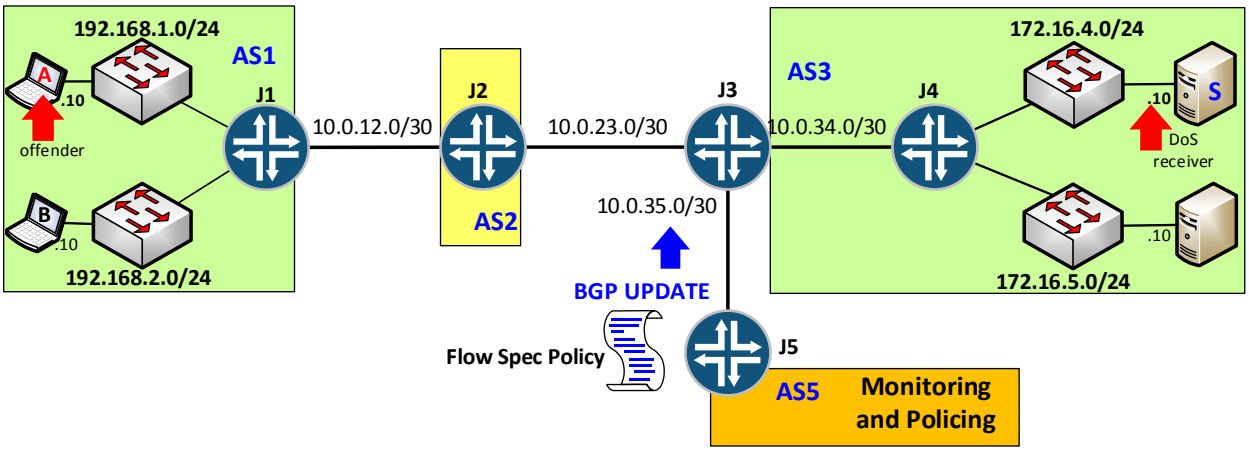


Figure 11. 76

This is the general idea. An offending host (192.178.1.10 in AS1) is attacking, with a denial of service, a server (172.16.4.10) located in AS3. In the meantime, Autonomous System AS5 is a monitoring and policy centre which provides special protection services to AS3. A monitoring tool (not shown) detects the attack and instructs the router J5 to send a remedy rule to block the traffic flow from 192.168.1.10 to 172.16.4.10. In this example, ICMP pinging will be use it to show the concept, but there are many rules and applications possible. This is just an introduction to FlowSpec.

Note: J3 is a Juniper MX-5i router with the capability, not all devices can do FlowSpec.

This is the configuration in router J5. First, the flow specification is created.

set system host-name J5 set routing-option router-id 10.5.5.5 set routing-options autonomous-system 5 edit routing-options **flow route** **BLOCK-ATTACK** set **match destination** 172.16.4.10/32 set **match source** 192.168.1.10 set match **protocol icmp** set **match icmp-type** echo-request set then **discard** set **term-order standard**	The routing option flow route creates a special entry in the local routing information base (rib) called inetflow. This flow route is bound to the conditions listed in the filter. In this case, a source and destination IPv4 address-pair and a protocol message (ICMP Echo Request).

Second, a routing policy is written in J5.

edit policy-options policy-statement **FLOW-FILTER** set term 1 from **rib inetflow.0** set term 1 then accept set term 2 then reject	This policy collects the filter object with its conditions from the RIB inetflow.

Last, the policy is called under the BGP configuration being applied to the peer J3.

edit **protocol bgp** group TO_AS3 set type external set neighbor 10.0.35.1 peer-as 3 set family inet unicast set **family inet flow** set **export** **FLOW-FILTER**	The special address family AFI is specified to activate the capability in the BGP messages. An export matches the name of the policy that will be advertising the filter rules to the neighbor J3.

This is the configuration of J3 pertaining only to the BGP peering with J5.

```
set system host-name J3
set routing-options router-id 10.3.3.3
set routing-options autonomous-system 3

edit protocol bgp group FROM_AS5
   set type external
   set neighbor 10.0.35.2 peer-as 5
   set family inet unicast
   set family inet flow no-validate FlowSpec-FILTER

edit policy-options policy-statement FlowSpec-FILTER
   set term 1 from neighbor 10.0.35.2
   set term 1 then accept
```

The Flow Spec Filter would have no effect if the receiving peer does not implicitly accept it.

The policy FlowSpec-FILTER accepts the filter rules sent by J5.

J3 is enabled, under the BGP group, to accept, without question, the flow specifications.

Once that the configuration is completed, router J5 sends the following OPEN message advertising its FlowSpec capabilities.

Border Gateway Protocol
Marker FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
Length: 38 bytes
TYPE: (number 1) OPEN Message

My BGP version is 4
My Autonomous System Number is 5
My Hold Time is 90 seconds
My BGP identifier is **10.5.5.5 (this is J5)**
Optional Parameters Length = 38 bytes
Optional parameters:
Multiprotocol Extensions Capability
Address Family Identifier **AFI number 1 = IPv4**
Subsequent AFI **(SAFI) = Flow Spec Filter (133)**

A BGP OPEN message describes the support for **IPv4 SAFI 133** which is **Flow Specification Filter**.

Once again, the BGP protocol has been expanded with a new capability.

Figure 11. 77

This is the BGP UPDATE (edited) message from J5 to J3:

Border Gateway Protocol
Marker FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
Length: 68 bytes
TYPE: (number 2) UPDATE Message

Withdrawn routes Length: 0
Path attributes:
ORIGIN IGP
AS_PATH: 5
EXTENDED COMMUNITIES:
Community AS Flow Spec Traffic-rate
Transitive, Optional, Complete
Experimental (80_H) Flow Spec traffic rate (06_H)
Community: 0 Rate shaper: 0

Attribute MP_REACH_NLRI
FLOW_SPEC_NLRI
Address Family AFI IPv4 (1)
Subsequent AFI: Flow Spec Filter (133)

FLOW_SPEC_NLRI:
Filter: Destination prefix (172.16.4.10/32)
Filter: Protocol
Filter: ICMP type = 8

A community is automatically appended to the UPDATE. This community is specifically designed for Flow Spec.

The Multiprotocol NLRI advertises address family AFI and SAFI for Flow Spec Filter.

Finally, the filter rule is exported in the field NLRI. The destination prefix is accompany by the protocol and type of message that is being controlled.

Figure 11. 78

As a consequence, the routing table in J3 shows a special table called inetflow with the entry indicating that the security rule has been accepted.

```
felix@J3> show route table inetflow.0
192.168.1.10, 172.16.4.10, proto = 1, icmp-type = 8 /term: 1
* [BGP/170], localpref 100, from 10.0.35.2, AS_Path 5 i Fictitious
```

Also, the MX-5i router J3 has firewall capacity. When the firewall get tested by pinging (ICMP-type 8) the firewall counter shows the activity while the matching traffic get blocked. On the other hand, any other traffic that is not filtered goes through.

```
felix@J3> show firewall
Filter_flowspec_default_inet_
Name:
192.168.1.10, 172.16.4.10, proto = 1, icmp-type = 8
Bytes Packets
 6000    4
```

Pictorially, the DoS attack directed from 192.168.1.10 to 172.16.4.10 would be blocked at the border router J3.

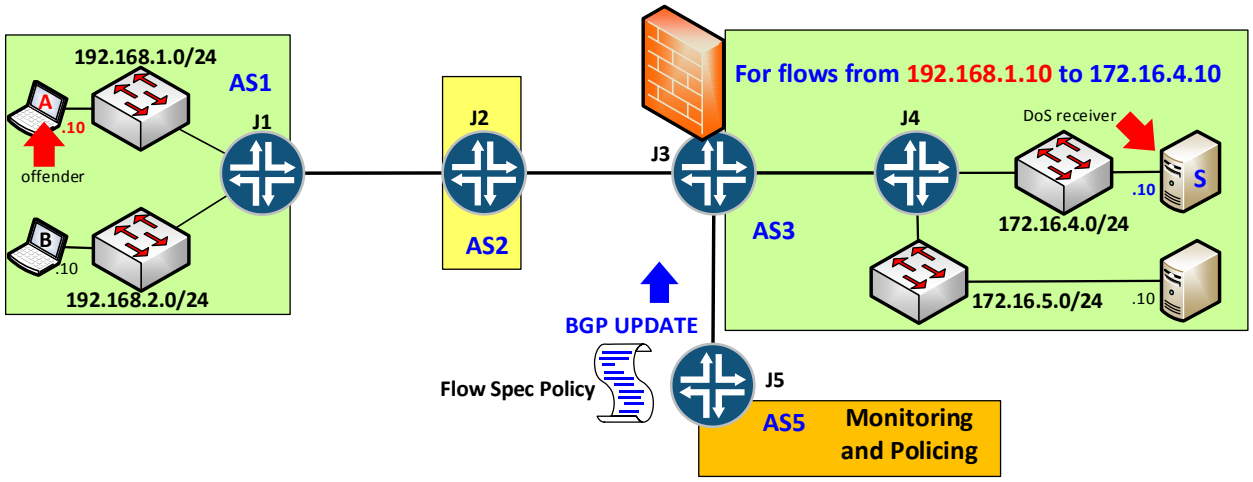


Figure 11. 79

This is just a sample demonstration of the kind of function that FlowSpec can provide. There are more options such as controlling TCP segments, at the level of protocol flags inclusive, control of flow rate, and more. It all works thanks to the versatility of the BGP protocol headers that have been expanded to support this application.

Chapter References

[1] E. Rossen. Y. Rekhter. RFC- 4364 BGP/MPLS IP Virtual Private Networks (VPNs). February 2006.

[2] K.Kompella, B.Kothari, R. Cherukuri. RFC 6624 Layer 2 Virtual Private Networks Using BGP for Auto-Discovery and Signaling. May 2012.

[3] K. Kompella, Y. Rekhter. RFC 5462. Virtual Private LAN Service Using BGP for Auto-Discovery and Signaling. January 2007.

[4] Sajassi, R. Aggarwal, N. Bitar, A. Isaac. J. Uttaro. J. Drake. W. Henderickx, RFC-7432 BGP MPLS Based Ethernet VPN. February 2015.

[5] P. Marques, N. Sheth, R. Raszuk, B. Greene, J. Mauch, D. McPherson. RFC-5575 Dissemination of Flow Specification Rules. August 2009.

CHAPTER 12

BGP in the Data Centre

If there is an infrastructure that has experimented the most significant changes in so few years, that is the data centre. From a traditional storage facility, the data centre has evolved into the focal point for content and application delivery for the Internet. The concept of cloud computing has the data centre as its main supporting axis and virtualization as its main tool. Nevertheless, network fundamentals remain the same. Servers have to be reachable, data has to be carried from place to place and all of that needs signaling protocols. Remarkably, BGP with its flexible capabilities is well suited to act as an advertiser for services in the data centre. In this chapter, two recent applications of BGP are explained. The first one is BGP in the Leaf and Spine switching structure and the second is the use of BGP as a signaling protocol to advertise physical addresses.

BGP in the Leaf and Spine Topology

The following diagram shows a typical Clos arrangement for a data centre that consists of three layers, the spine, the leaves and the compute from top to bottom.

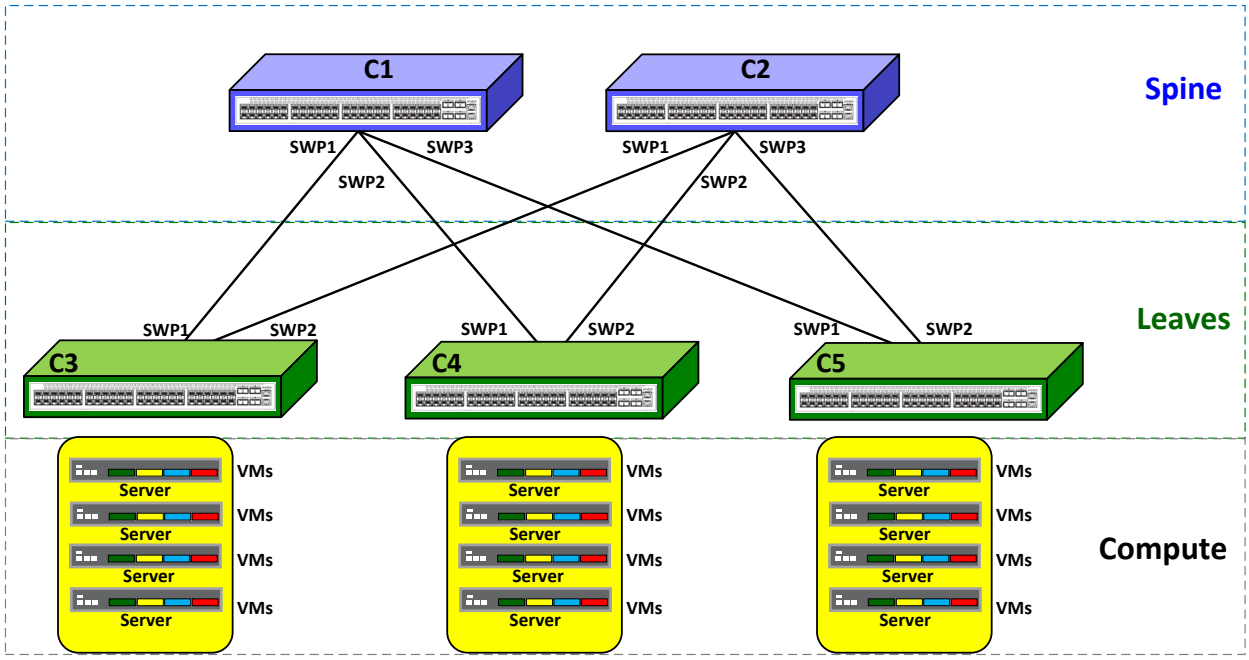


Figure 12.1

Leaf and spine switching structures are inspired in the research [1] done by Charles Clos in 1952 for telephone exchanges. In the Clos design, there would always be a path between an idle input port and an idle output port across regardless of the traffic load, hence the whole structure becomes a non-blocking switch. Basically, as long as there are available source and destination ports, the switching structure can find a path across.

Although the original Clos switch was built with electromechanical relays, his theory has found application in modern data centres. In fact, by arranging high performance LAN switches in the manner shown above, a Clos switching fabric is obtained with the same non-blocking capacity.

In the diagram, the bottom edge or compute, represent racks with physical servers where multiple virtual machines (VM) are orchestrated. The traffic of these virtual servers is aggregated via top of the rack (TOR) devices that integrate to the leaf layer. The top layer or spine interconnects several leaves to ensure that any destination can be reached.

In the diagram, the devices are not shown with the traditional router icon for a reason. These are generic switches that support open operating systems instead of a vendor specific OS. Therefore, these switching devices should be seen as "general purpose boxes" with the ability of being configured with software defined tools. The terms "white box" and "bare-metal" are used in the data centre's jargon to designate these devices. In principle, they can behave in multiple ways as routers, switches, firewalls and more depending of the needs. For example, this particular leaf and spine Clos topology has been implemented with generic switches with the Operating System Linux (Cumulus).

BGP behaviour in the Leaf and Spine Topology

The traffic interest in the data centre is aimed at reaching the many virtual machines located in virtual networks. There are different solutions to advertise the presence of those networks and devices including the use of routing protocols. Among routing protocols, BGP has advantages that make it a desirable candidate for propagating routing knowledge in the data centre.

- First, BGP is truly **Multiprotocol**, so it is not restricted to just IPv4 addresses, it can advertise IPv6 routes, VPN routes of different types, even MAC addresses. BGP can do that by including **Address Family Information AFI** in its **Update** messages. For example, all of these are address families: IPv4 Unicast, IPv4 Multicast, IPv6 Unicast, IPv6 Multicast, Layer 3 VPN, and Layer 2 VPN.

- Second, BGP can be adapted to new applications. Seen from an abstracted view, BGP just carries **objects with attributes**. As long as the objects and their properties are understood by all routers, that knowledge is spread around. This is possible, because BGP routers announce their **capabilities** to other routers within the OPEN messages. New capabilities have been added to BGP over the years.

- Third, BGP is the protocol that keeps the whole Internet connected. All data centres have gateway routers that are running BGP already and extending its **scope** into the data centre eliminates the need to redistribute the routes from an IGP into BGP or to inject the routes manually.

- Fourth, BGP has a well-documented, feature rich, **policy engine** to control and manipulate routing at the will of the network engineer.

- Fifth, BGP has a large **scale capacity** to handle hundreds of thousands of prefixes. No other routing protocol comes even close to that.

- Finally, BGP represents **low overhead** traffic to the network once it has converged because BGP only updates the changes in the topology as they happens. If there are no changes, BGP just sends small KEEPALIVE messages periodically. In contrast, all the other routing protocols frequently re-advertise their whole routing information.

For all these reasons, it makes sense to run this protocol inside the data centre. Nevertheless, BGP is not an IGP and it must not be configured to act as such. In a traditional implementation, when BGP is running inside a routing domain (iBGP) is only to get routes across from edge to edge. Inside the routing domain an IGP is needed anyways to build the internal IP connectivity.

In the data centre, flexibility is the key word because new services must be spun up as quickly as possible by a combination of virtual orchestration and efficient physical deployment. For example, a new server is deployed, many virtual machines are created on top, and optical fibre is strung up to interconnect with other devices and that should be up and running in the shortest time. However, if this new installation takes place over a rigid network underlay, then several additional configuration tasks will still be needed like configuring static IP addresses, re-configuring routing protocols and others. That slows down the overall process.

Fundamentally, there is the need to simplify the configuration tasks of IP connectivity and to avoid a rigid underlay network. This brings up the main objective of the model being analyzed which is how to make BGP as self-configurable as possible with a minimum configuration of the underlay network.

Significantly, BGP is the only routing protocol that does not have a built-in neighbour discovery by default. This was a design decision from the very beginning [2] of the protocol. BGP is transported by TCP and that implies that a reachable IP address is required to open a connection. This destination IP address is supplied by the fingers of the network engineer when the protocol is configured on a router. Basically, peering is a deliberate action by the administrators. This paradigm makes sense when the peering is between an Internet Service Provider and a customer network. In that case, the ISP provides a direct point to point link with fixed addresses to the client. The BGP configuration files are very specific about having those addresses. Otherwise, BGP sessions cannot be established. But this conventional BGP peering model is inadequate for the data centre. Instead, a different model that minimizes the configuration tasks is needed. That model would ideally accomplish the following:

- Have any interface ready to be used for BGP peering. Basically, if a new physical link is needed, then plug it in and let it go without having to statically assign unique IP addresses to each end point.

- Allow BGP routers to self-discover neighbours. That is to reduce the need to be reconfiguring BGP every time that there is a new neighbour.

To solve the first point, an IP address is assigned to the loopback interface of each white box and then every other interface will also use the same address. This might sound blasphemous to a network engineer. In any other device, doing this would be rejected as the configuration mistake known as address overlap. But this is, again, a different model of doing networking. The following diagram shows the configuration file of Cumulus Linux [3] running in the bare-metal Dell switch named C1.

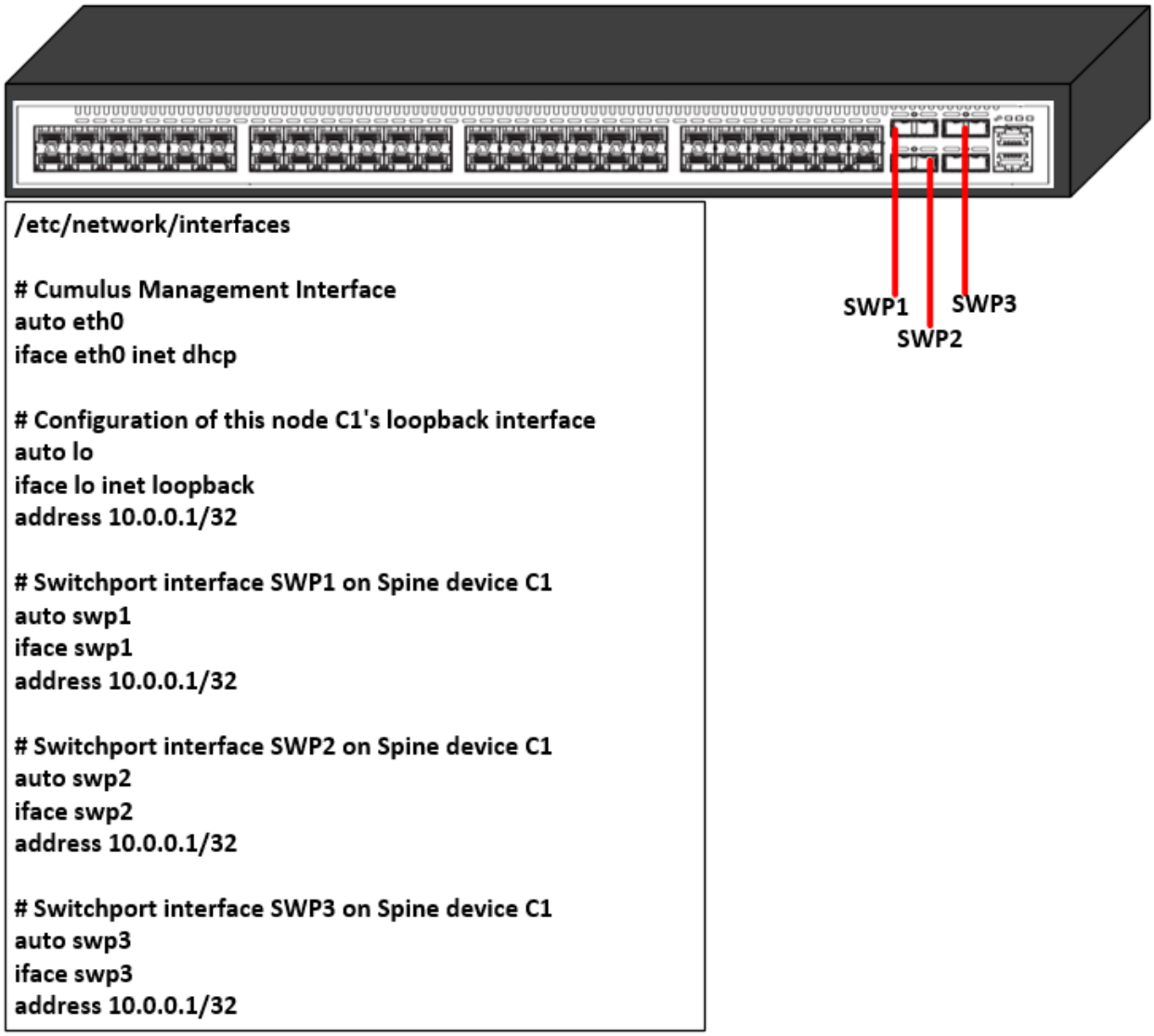


Figure 12.2

The way to see this configuration is that the address belongs to the node and the interfaces are just pathways to get to the same destination. When a BGP router function is enabled in the node, it would use such address to peer to another. This is basically the router identification number needed for peering.

Notice that all the physical interfaces respond to the loopback address. That is particular to this implementation. Once that this node's IPv4 address is configured, it needs to be advertised. Peering with loopback addresses is a well-known practice when deploying BGP inside autonomous systems or iBGP. However, in that case, the loopback addresses are advertised first by another routing protocol, an IGP like OSPF or IS-IS for example. But deploying another routing protocol, just to run BGP in the data centre is unwanted. There must be a solution to this circular problem: how can the loopback address be advertised without using an IGP and without having to configure unique addresses per each interface?

There is a simple protocol already designed to do just that. It is **Link Layer Discovery Protocol LLDP** [4]. This protocol runs encapsulated by Ethernet and it advertises basic information representing the sender device. Therefore, this takes care of one problem already.

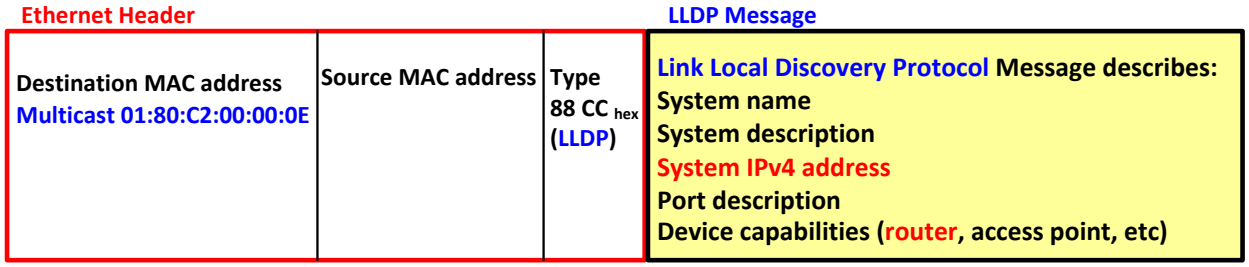


Figure 12.3

The second item in this configuration model is to make BGP as self-configurable as possible while avoiding static configuration of the underlay network and running an IGP. To better visualize the problem to be solved, it helps to have a look at the encapsulation of BGP in the next diagram:

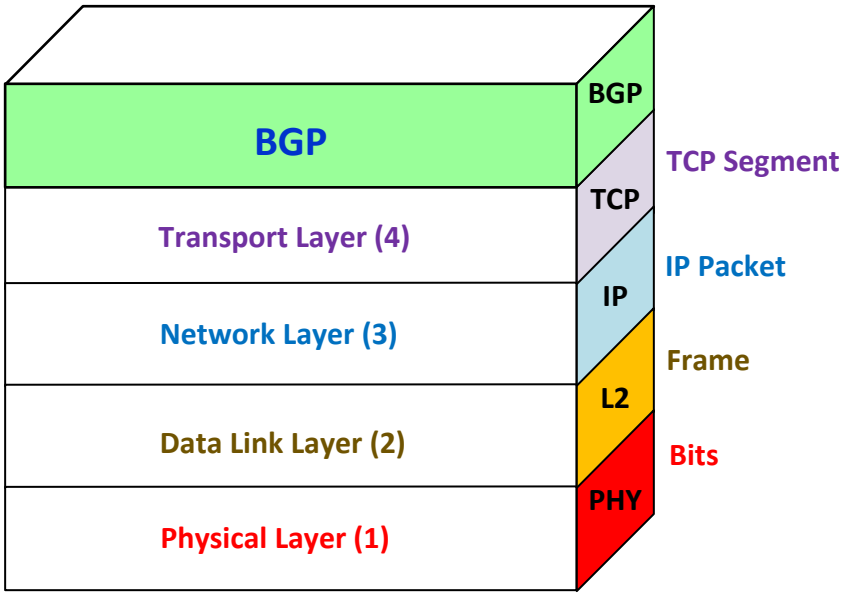


Figure 12.4

The encapsulation of BGP requires that either IPv4 or IPv6, be present in the network. In contrast, the goal of the current implementation is to allow any interface to be used without setting up IP addresses specifically. If only the interfaces would self-configure with a layer 3 address, that would solve this problem!

IPv4 cannot do that, but IPv6 can because IPv6 has the feature of self-configuring Link Local Addresses. The following diagram shows the spine device C1 self-configuring an **IPv6 Link Local Address** for its interface SWP1. Basically, the MAC address of the interface is modified and embedded onto an IPv6 address.

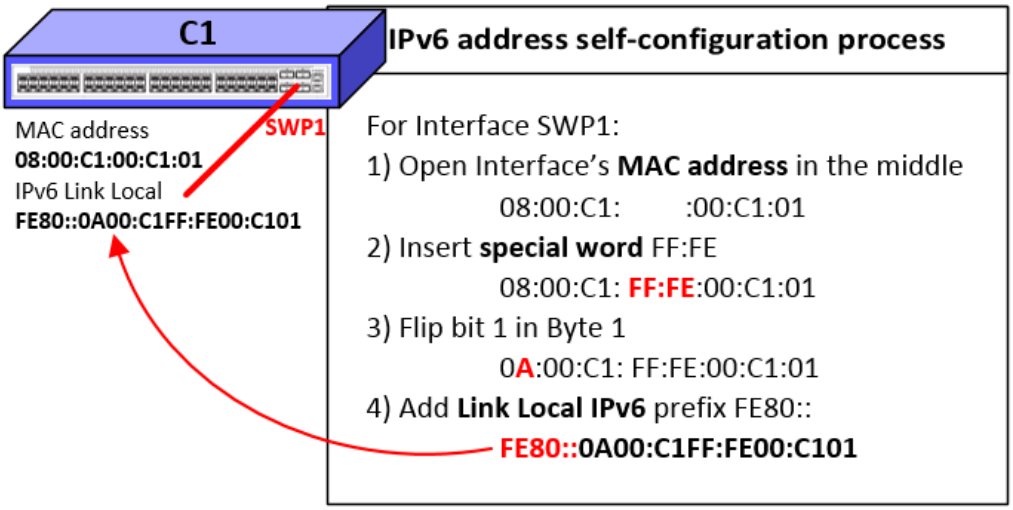


Figure 12.5

Furthermore, IPv6 [5, 6] has local advertising capabilities using the **ICMPv6** protocol. In principle, once that the devices have a self-configuring address per interface, they could start a session to the IPv6 address advertised from the other side of the link.

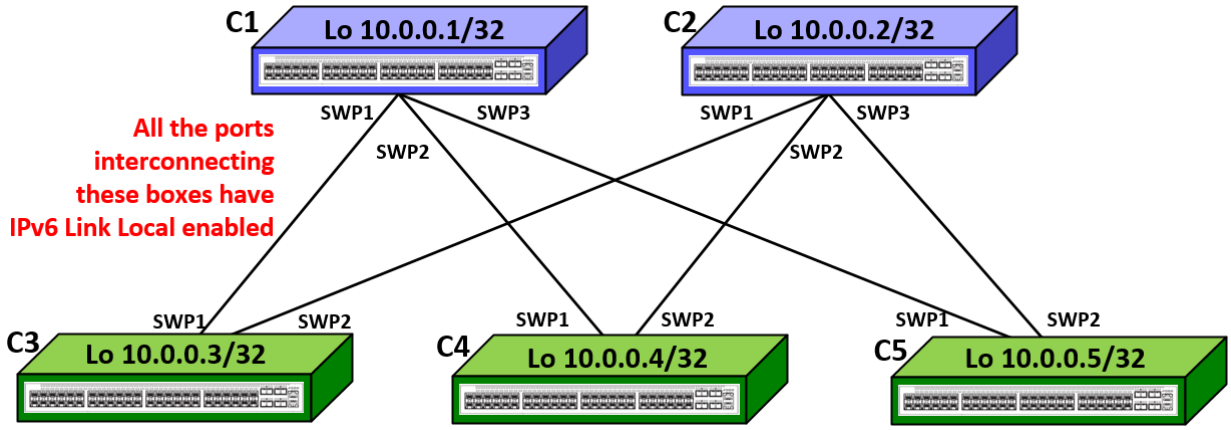


Figure 12.6

However, this brings up yet another problem. BGP can advertise routes from different address families, but for that knowledge to be used [2] to actually route packets, a **Next Hop Address** of the **same address family** must be present in the routing table. Basically, a BGP router would know the prefix route but there would be no valid next-hop to send the traffic forward. That would imply that if the services in the data centre had IPv4 addresses, their presence could be advertised by BGP all right, but they could not be used without IPv4 being configured in the interfaces of the network. Yet another circular problem that needs a solution.

There is a solution to this dilemma as well. The standard document RFC 5549 [7] specifies that the Next Hop attribute of an IPv4 route advertised by BGP, does not necessarily have to be another IPv4 prefix, but it could be an IPv6 prefix instead. Fundamentally, RFC-5549 relaxes the default BGP rule about Next-Hop so it can be used in this type of applications.

Once that there is a path to communicate, TCP can establish a transport session. When this process completes, BGP neighbors send OPEN messages to each other. Inside this type of message, there is a field with the router-id which is the loopback address plus other fields that contain the description of the capabilities.

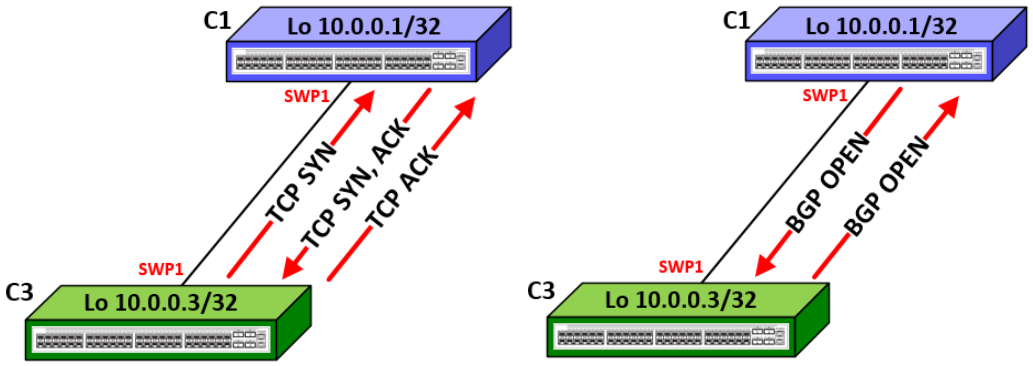


Figure 12.7

The routers then proceed to exchange routing information within UPDATE messages. Because of the modification in the standard behaviour of the BGP protocol, the IPv6 Link Local address can be used as the Next Hop for IPv4 routes. This process is analyzed in detail (with data sniffing) later in this document.

When the configuration is completed, each bare-metal device becomes a router in a BGP Autonomous System as shown in the following figure.

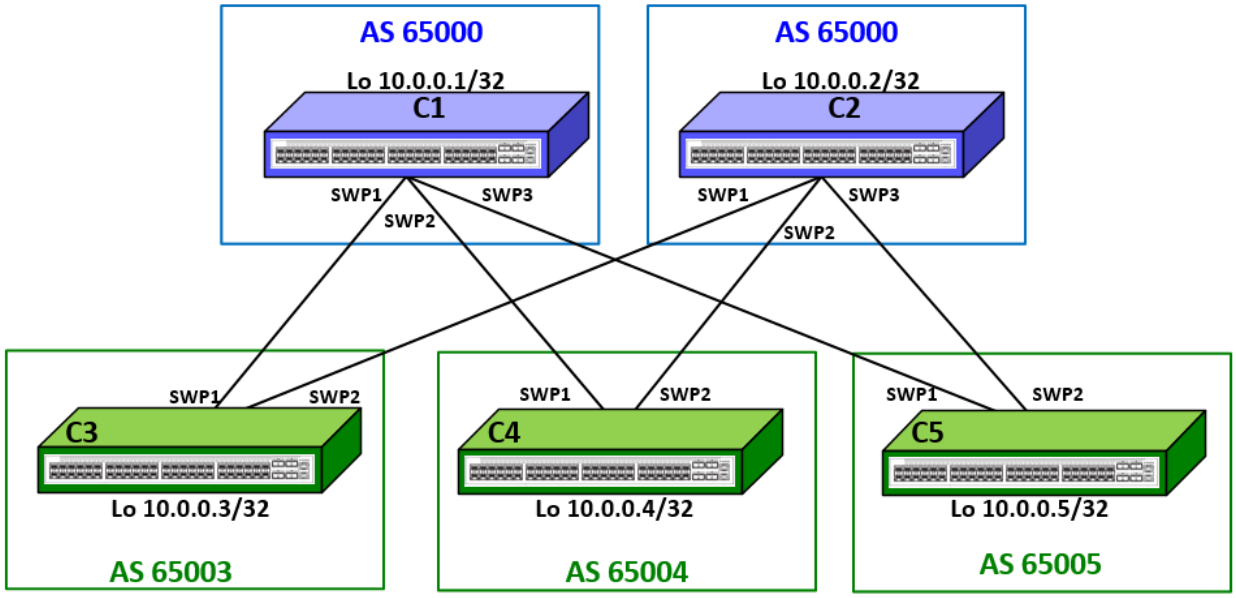


Figure 12.8

The autonomous systems can be seen as routing domain clouds from an abstract viewpoint. It is important to remark that in this BGP model, the spine has two clouds with the same AS number 65000 even though they are not directly connected. That is intended to be so.

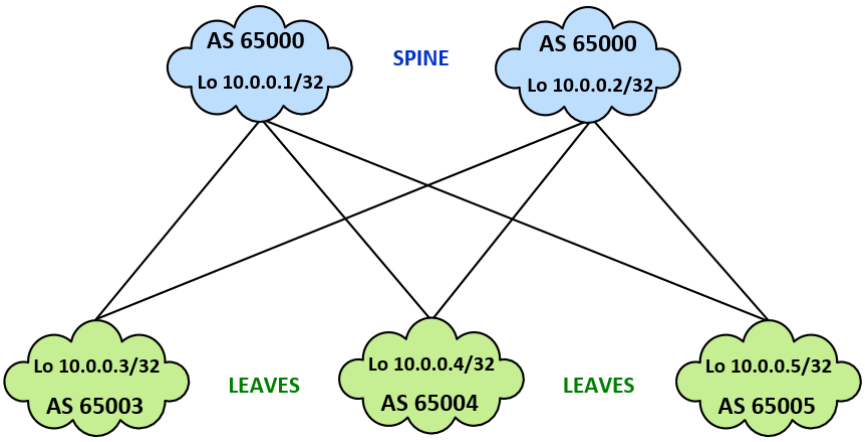


Figure 12.9

For the BGP purist, this immediately invokes a violation of the BGP rule that says that these two parts of AS 65000 should be linked by (internal) iBGP to integrate the same autonomous system. However, that is not so in this model. The spine routers do not talk iBGP to each other because simply put, they do not need to. This is an interesting concept that deserves analysis to understand this deployment model.

Assuming that AS 65005 advertises prefix 172.16.5.0/24 to both AS 65000s in the spine of the topology. The prefix will be relayed by both AS 65000s to the Leaves AS 65003 and AS 65004. That accomplishes the main objective of the whole setting; that the leaves get to know the prefixes located in the compute layer.

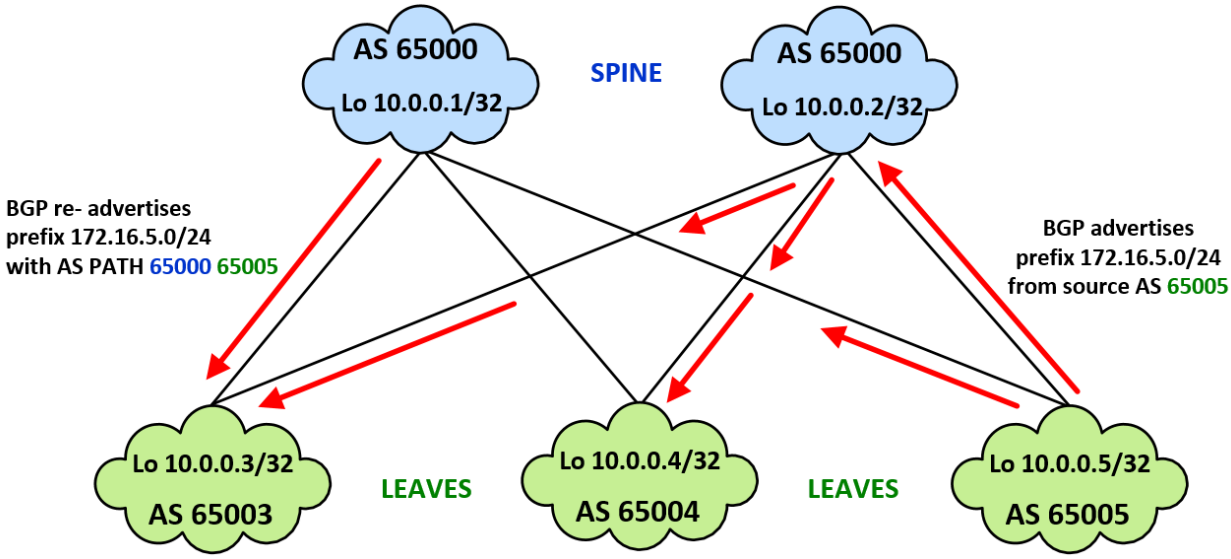


Figure 12.10

Now, one leaf AS is chosen to simplify the explanation in the following figure.

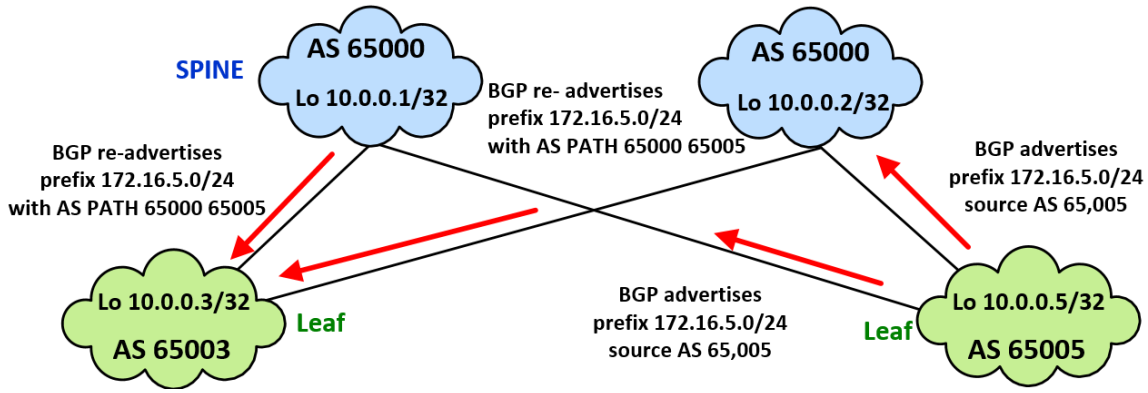


Figure 12.11

From the viewpoint of AS 65003, there is only one AS65000. Basically, AS 65003 has no means to know that the AS 65000 is physically split. In consequence, the leaf router in AS 65003 has two valid equal paths, installed in its BGP database, to the destination prefix 172.16.5.0/24. BGP carries the Autonomous System numbers that the routing updates have traversed in the **BGP Attribute AS-PATH**. In this case, the AS-PATH indicates that the route can be reached thru AS 65000 and the destination is located inside AS 65005.

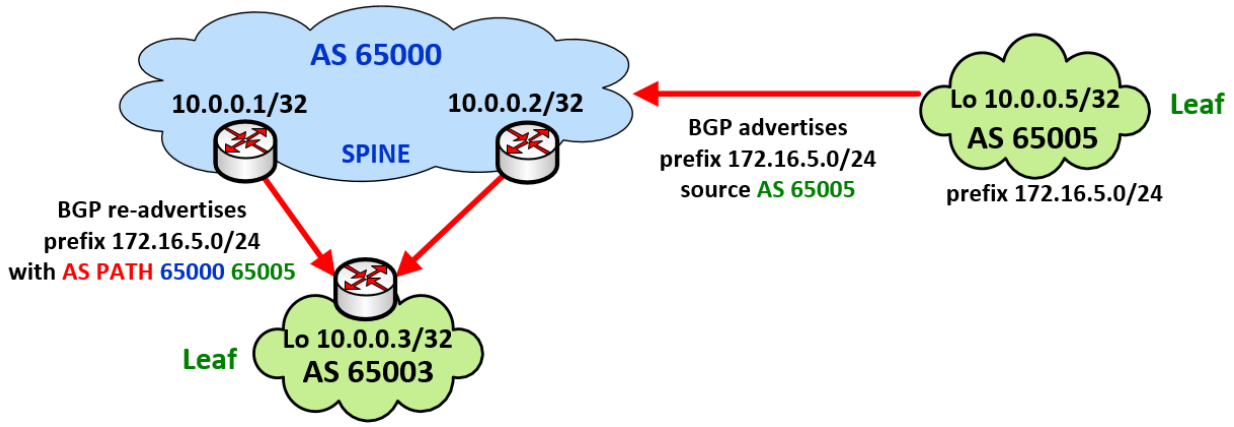


Figure 12.12

From the view point of C3, the path toward the destination 172.16.5.0/24 looks like this in the BGP routing database:

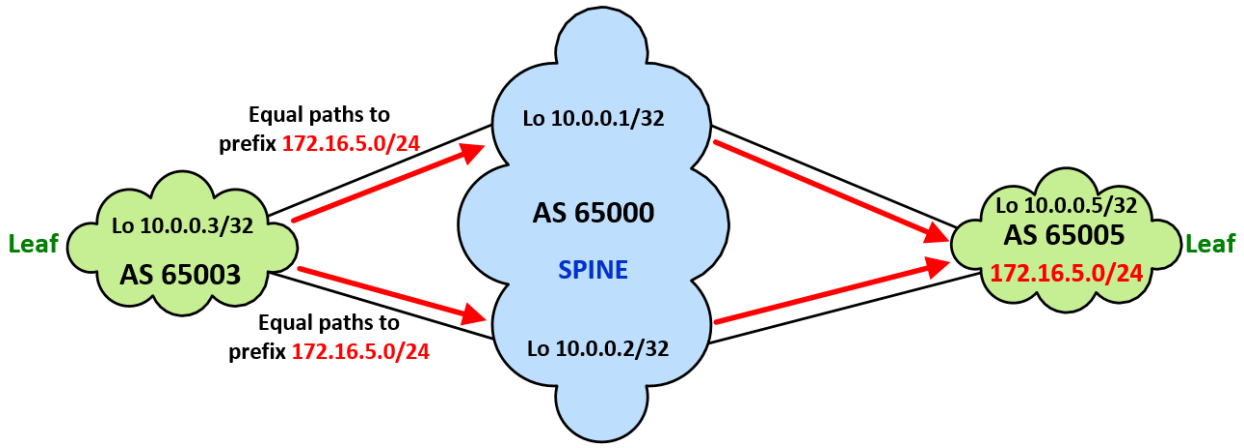


Figure 12.13

In principle, having two equal paths is a good thing if the routing protocol can do load balancing to equally distribute the traffic using both interfaces. However, the default behaviour of BGP is to run an algorithm to choose the best path to be presented to IP for its inclusion in the routing table. If all the factors considered are equal, the best path algorithm reaches a tie-breaker and then chooses one path. That would leave the other path idle and that represents a waste of a high speed link and even worst, the disabling the Clos non-blocking functionality of the leave and spine topology.

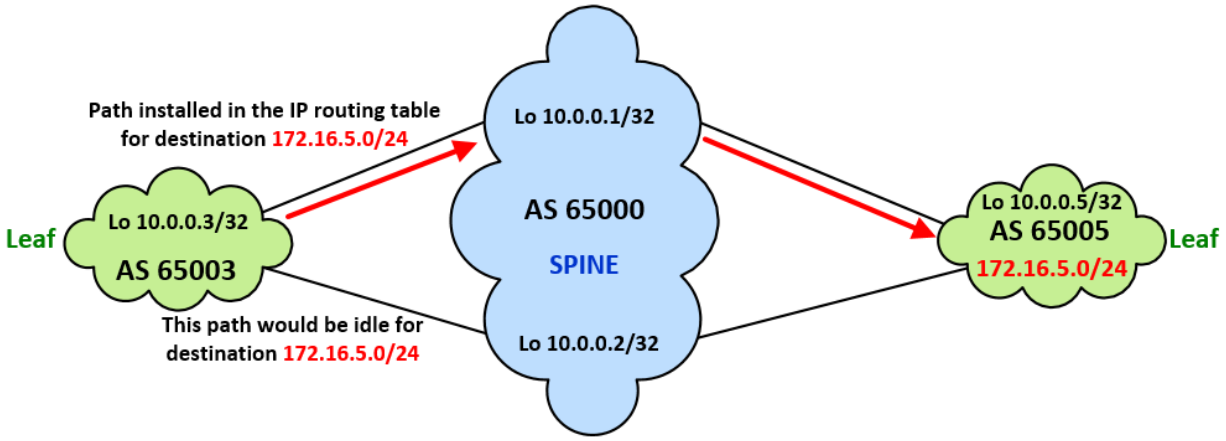


Figure 12.14

To force BGP to present both paths to the IP protocol for inclusion into the routing table as equally valid, BGP has to be tweaked with a special command that relaxes the BGP default behaviour.

```
bgp bestpath as-path multipath-relax
```

The solution is that the IP routing table has installed two equal cost paths to the destination.

```
C3# show ip route
C> * 10.0.0.3/32 is directly connected, lo
C * 10.0.0.3/32 is directly connected, swp1
C * 10.0.0.3/32 is directly connected, swp2
C> * 172.16.3.0/24 is directly connected, swp3
B> * 172.16.4.0/24 [20/0] via FE80:: 0A00:C1FF:FE00:C101, swp1
   * 172.16.4.0/24          via FE80:: 0A00:C2FF:FE00:C202, swp2
B> * 172.16.5.0/24 [20/0] via FE80:: 0A00:C1FF:FE00:C101, swp1
   * 172.16.5.0/24          via FE80:: 0A00:C2FF:FE00:C202, swp2
```

Table 12.1

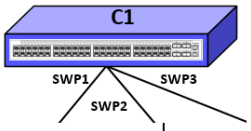
Configuration of BGP in the Quagga Router

Cumulus Linux supports the open source router Quagga. This is a software suite [10] that includes several routing protocols including BGP. Its initial configuration is done in a typical Linux manner with text file editing. The configuration of spine node C1 is explained next while the complete configuration of all nodes can be found in the appendix of this document. First, the routing daemons file is edited to enable BGP:

```
/etc/quagga/daemons

zebra=yes
bgpd=yes
(Everything else stays in no.)
```

Second, the quagga router is configured to support the BGP model of peering to the loopbacks while the interfaces are just running on IPv6 Link Local.

```
/etc/quagga/Quagga.conf

# Configuration File for Quagga Router.
# This is Cumulus Linux node C1 spine.
!
interface swp1
no ipv6 nd suppress-ra
ipv6 nd ra-interval 3
!
interface swp2
no ipv6 nd suppress-ra
ipv6 nd ra-interval 3
!
interface swp3
no ipv6 nd suppress-ra
ipv6 nd ra-interval 3
!
```



Each interface has the same two commands:

- By default, all the interfaces have IPv6 neighbour discovery (nd) disabled.
- That is changed with the command "**no ipv6 nd suppress-ra**" which enables IPv6 nd so the device starts sending **ICMPv6 Router Advertisement** (ra) Messages.
- The command "**ipv6 nd ra-interval 3**" just sets the message interval to every 3 seconds.

Figure 12.15

The file continues with the BGP configuration.

/etc/quagga/Quagga.conf

```
router bgp 65000
bgp router-id 10.0.0.1
```

This router is located at Autonomous System 65000 and its router-id is the loopback address 10.0.0.1/32.

```
maximum-paths 64
```

This implementation supports up to 64 equal cost paths to a destination prefix.

```
bgp bestpath as-path multipath-relax
```

The default BGP behaviour of choosing one best path is relaxed and multiple paths are accepted.

```
neighbor LEAF peer-group
```

A BGP peer group is defined

```
neighbor LEAF remote-as external
```

In a conventional BGP configuration, the remote autonomous system is required. In this BGP model, any external AS number is accepted.

```
neighbor LEAF advertisement-interval 0
neighbor LEAF timers 1 3
```

KEEPALIVE timer is set to be very frequent (1)
HOLDTIME is set to 3 seconds or 3 missed KEEPALIVES

```
neighbor LEAF capability extended-nexthop
```

The capability extended nexthop is enabled and BGP will accept IPv6 next-hops for IPv4

```
neighbor swp1 interface
neighbor swp1 peer-group LEAF
neighbor swp2 interface
neighbor swp2 peer-group LEAF
neighbor swp3 interface
neighbor swp3 peer-group LEAF
```

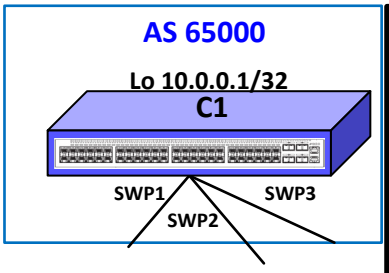
In a conventional BGP configuration, the BGP neighbour address is required. In this BGP model, the neighbour is any peer located via the interface.

```
address-family ipv6 unicast
neighbor LEAF activate
neighbor swp1 activate
neighbor swp2 activate
neighbor swp3 activate
exit-address-family
```

The BGP neighbours are activated under the IPv6 Unicast Address Family

Figure 12.16

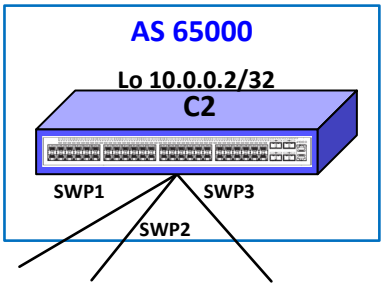
The results can be observed once that all bare-metal boxes have been completed. In effect, nodes C1 and C2 in the spine AS65000 can see all the prefixes located in the compute layer. The prefix 172.16.3.0/24 is located in AS65003, the prefix 172.16.4.0/24 is located in AS65004, and prefix 172.16.5.0/24 is located in AS65005.




```
C1# show ip route

C *  10.0.0.1/32 is directly connected, via swp1
C *  10.0.0.1/32 is directly connected, via swp2
C *  10.0.0.1/32 is directly connected, via swp3
C>* 10.0.0.1/32 is directly connected, lo0

B>* 172.16.3.0/24 [20/0] via FE80::0A00:C3FF:FE00:C301, via swp1
B>* 172.16.4.0/24 [20/0] via FE80::0A00:C4FF:FE00:C401, via swp2
B>* 172.16.5.0/24 [20/0] via FE80::0A00:C5FF:FE00:C501, via swp3
```

Figure 12.17

Both spine routers C1 and C2 have a similar view with all the destinations just one hop count away. Notice that for the routes advertised by BGP, the NEXT HOPs are the IPv6 Link Local addresses.

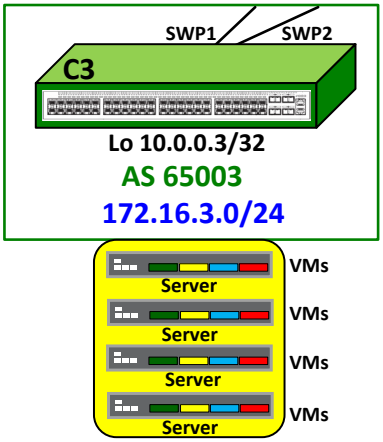



```
C2# show ip route

C *  10.0.0.2/32 is directly connected, via swp1
C *  10.0.0.2/32 is directly connected, via swp2
C *  10.0.0.2/32 is directly connected, via swp3
C>* 10.0.0.2/32 is directly connected, lo0

B>* 172.16.3.0/24 [20/0] via FE80::0A00:C3FF:FE00:C302, via swp1
B>* 172.16.4.0/24 [20/0] via FE80::0A00:C4FF:FE00:C402, via swp2
B>* 172.16.5.0/24 [20/0] via FE80::0A00:C5FF:FE00:C502, via swp3
```

Figure 12.18

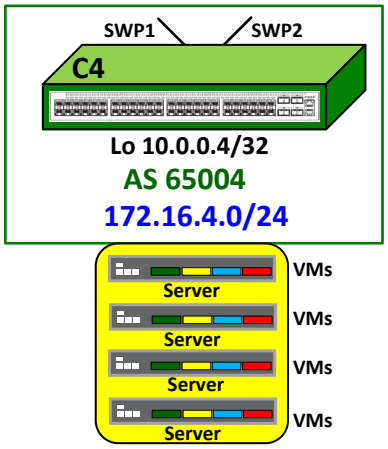
The routers located in the leaf layer have a different view than the spine layer routers. For each remote destination, the routing table of each router, C3, C4 or C5, has two equal cost paths. This demonstrate the ultimate objective of this topology which is providing the leaf nodes with equal redundant active paths to all the destinations in the compute layer.

In contrast, in a standard BGP deployment, each remote prefix would have only one entry in the IP routing table, the one chosen to be the best by the BGP best path decision algorithm. That is not the case in this Leaf and Spine model since the BGP algorithm has been relaxed (bgp bestpath as-path multipath-relax) to accept more than one path to the same destination prefix.




```
C3# show ip route
C *  10.0.0.3/32 is directly connected, via swp1
C *  10.0.0.3/32 is directly connected, via swp2
C>* 10.0.0.3/32 is directly connected, lo0
C>* 172.16.3.0/24 is directly connected, swp3

B>* 172.16.4.0/24 [20/0] via FE80::0A00:C2FF:FE00:C201, via swp2
B  * 172.16.4.0/24          via FE80::0A00:C1FF:FE00:C101, via swp1

B>* 172.16.5.0/24 [20/0] via FE80::0A00:C2FF:FE00:C201, via swp2
B  * 172.16.5.0/24          via FE80::0A00:C1FF:FE00:C101, via swp1
```

Figure 12.19




```
C4# show ip route

C *  10.0.0.4/32 is directly connected, via swp1
C *  10.0.0.4/32 is directly connected, via swp2
C>* 10.0.0.4/32 is directly connected, lo0
C>* 172.16.4.0/24 is directly connected, swp3

B>* 172.16.3.0/24 [20/0] via FE80::0A00:C2FF:FE00:C202, via swp2
B  * 172.16.3.0/24          via FE80::0A00:C1FF:FE00:C102, via swp1

B>* 172.16.5.0/24 [20/0] via FE80::0A00:C2FF:FE00:C202, via swp2
B  * 172.16.5.0/24          via FE80::0A00:C1FF:FE00:C102, via swp1
```

Figure 12.20

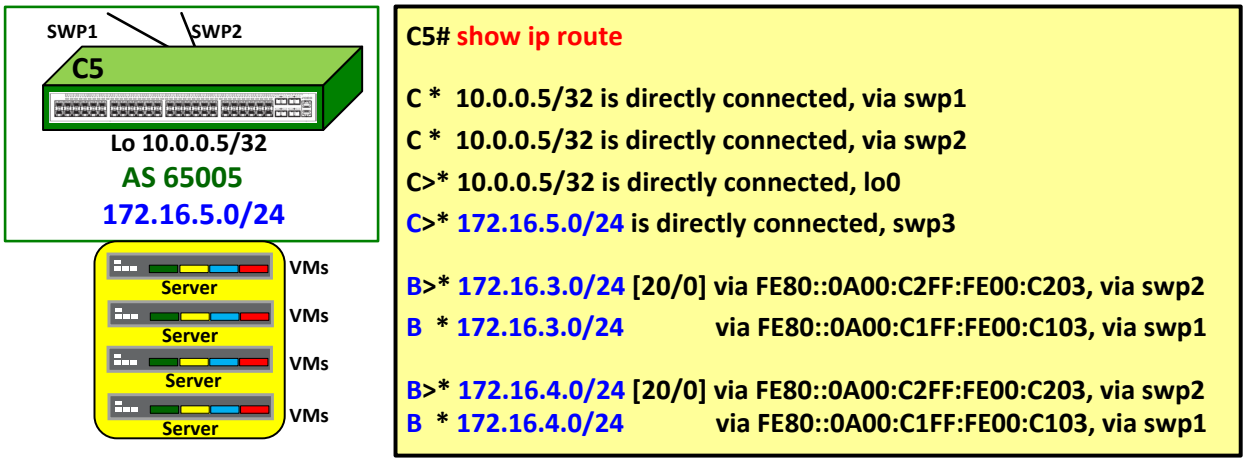


Figure 12.20

At this point, all the destinations routes are mutually reachable. Since the configuration has been completed then it is time to analyze the protocol messages exchanged between one spine node and one leaf node.

Analysis of transactions between nodes C1 and C3

A strategic location between spine node C1 and leaf node C3 has been selected to sniff the conversation between the two devices. The messages are explained from the very beginning.

The first message that appears is an **ICMPv6 Multicast Listener Report Message**. This is used by IPv6 able devices to discover if there are listeners on a specific interface. [5]

No.	Source	Destination	Protocol	Info
1	::	ff02::16	ICMPv6	Multicast Listener Report Message v2

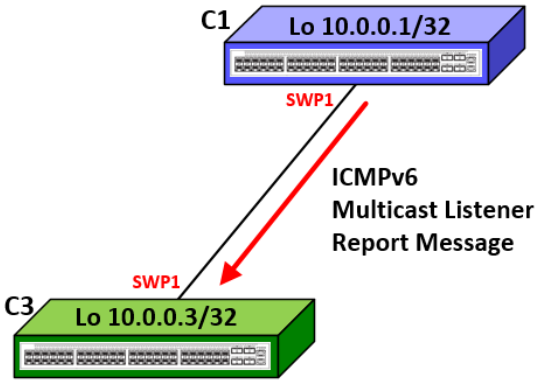


Figure 12.21

In this case, Cumulus box C1 is sending the message on its interface swp1. Observe the source MAC address in the snippet below. C1 has no clue about what is at the end of the link, so it sends to a destination IPv6 Multicast address hoping that there is something listening to that.

```
                    MAC address of C1's interface SWP1      Destination MAC Address for IPv6 Multicast
Ethernet II, Src: 08:00:c1:00:c1:01  Dst: IPv6mcast_16 (33:33:00:00:00:16)
Internet Protocol Version 6, Src: ::, Dst: ff02::16
Internet Control Message Protocol v6
  Type: Multicast Listener Report Message v2 (143)
  Code: 0
  Checksum: 0xab6e [correct]
  Reserved: 0000
  Number of Multicast Address Records: 2
▷ Multicast Address Record Changed to exclude: ff02::1:ff00:c101   Link Local of C1 SWP1
▷ Multicast Address Record Changed to exclude: ff02::2
```

Figure 12.22

The second message is an **ICMPv6 Neighbor Solicitation**. RFC-4861 [6] states that this type of message is sent when a node wants to know the address of another node connected to the same link. In the process, a node provides its own Link Layer Address to the target. This message is being sent to a multicast MAC address since C1 does not know the target node yet.

Source IPv6	Destination IPv6	Message	Link Local IPv6 Address for C1 interface SWP1
::	ff02::1:ff00:c101	ICMPv6 Neighbor Solicitation for fe80::a00:c1ff:fe00:c101	

This message is advertising a locally generated IPv6 address formed with the prefix FF80 and the modification of the MAC address of C1 swp1 interface. This is a **Link Local Address** and if another device is listening, it will learn such address.

```
Ethernet II, Src: 08:00:c1:00:c1:01 Dst: IPv6mcast_ff:00:c1:01 (33:33:ff:00:c1:01)
Internet Protocol Version 6, Src: ::, Dst: ff02::1:ff00:c101
Internet Control Message Protocol v6
  Type: Neighbor Solicitation (135)
  Code: 0
  Checksum: 0x3023 [correct]
  Reserved: 00000000
  Target Address: fe80::a00:c1ff:fe00:c101   Link Local IPv6 address
                                             of C1's interface SWP1
```

Figure 12.23

In the meantime, Cumulus C1 keeps sending Multicast Listener Report messages.

```
fe80::a00:c1ff:fe00:c101   ff02::16          ICMPv6      Multicast Listener Report Message v2
```

Next, C1 sends a different protocol message. This is a **Link Layer Discovery Protocol LLDP**. It advertises properties such as the node name, the interface name and its capabilities.

```
Ethernet II, Src: 08:00:c1:00:c1:01  Dst: LLDP_Multicast (01:80:c2:00:00:0e)
Link Layer Discovery Protocol
  ▷ Chassis Subtype = MAC address, Id: 08:00:27:f2:9d:0c
  ▷ Port Subtype = Interface name, Id: swp1
  ▷ Time To Live = 120 sec
  ▷ System Name = C1
  ▷ System Description = Cumulus Linux
  ▷ Capabilities
  ▷ Management Address
  ▷ Port Description = swp1
  ▷ End of LLDPDU
```

Figure 12.24

The following snippet shows the details of the LLDP message. It is interesting to observe that the message describes the device as being router capable and it also advertises the loopback address 10.0.0.1. Any other node listening to this message can learn this information now.

```
Capabilities
    .... .... .... ...0 = Other: Not capable
    .... .... .... ..0. = Repeater: Not capable
    .... .... .... .1.. = Bridge: Capable
    .... .... .... 0... = WLAN access point: Not capable
    .... .... ..1. .... = Router: Capable
    .... .... ..0. .... = Telephone: Not capable
    .... .... .0.. .... = DOCSIS cable device: Not capable
    .... .... 0... .... = Station only: Not capable
◂ Enabled Capabilities: 0x0010
    .... .... .... ...0 = Other: Not capable
    .... .... .... ..0. = Repeater: Not capable
    .... .... .... .0.. = Bridge: Not capable
    .... .... .... 0... = WLAN access point: Not capable
    .... .... ..1. .... = Router: Capable
    .... .... ..0. .... = Telephone: Not capable
    .... .... .0.. .... = DOCSIS cable device: Not capable
    .... .... 0... .... = Station only: Not capable
Management Address
  0001 000. .... .... = TLV Type: Management Address (8)
  .... ...0 0000 1100 = TLV Length: 12
  Address String Length: 5
  Address Subtype: IPv4 (1)
  Management Address: 10.0.0.1
  Interface Subtype: ifIndex (2)
  Interface Number: 1
  OID String Length: 0
Port Description = swp1
```

Figure 12.25

ICMPv6 Router Advertisement RA are messages that announce the presence of an IPv6 capable router. RA messages are periodically sent out of IPv6 configured interfaces to the All IPv6 Routers Address destination (FF02::1). Other devices obtain basic information about the router by listening to this multicast address. The following snippet shows that Cumulus box C1 (spine) is sending an RA message advertising the MAC address of the switchport (spw1) interface.

```
Ethernet II, Src: 08:00:c1:00:c1:01 Dst: IPv6mcast_01 (33:33:00:00:00:01)
Internet Protocol Version 6, Src: fe80::a00:c1ff:fe00:c101, Dst: ff02::1
Internet Control Message Protocol v6
  Type: Router Advertisement (134)
  Code: 0
  Checksum: 0x1f1a [correct]
  Cur hop limit: 64
▷ Flags: 0x00
  Router lifetime (s): 1800
  Reachable time (ms): 0
  Retrans timer (ms): 0                    MAC of C1's interface SWP1
▷ ICMPv6 Option (Source link-layer address : 08:00:c1:00:c1:01)
```

Figure 12.26

After this point the Router Advertisement message is repeated periodically:

Source	Destination	Message
fe80::a00:c1ff:fe00:c101	ff02::1	ICMPv6 Router Advertisement from 08:00:c1:00:c1:01
fe80::a00:c1ff:fe00:c101	ff02::1	ICMPv6 Router Advertisement from 08:00:c1:00:c1:01
fe80::a00:c1ff:fe00:c101	ff02::1	ICMPv6 Router Advertisement from 08:00:c1:00:c1:01
fe80::a00:c1ff:fe00:c101	ff02::1	ICMPv6 Router Advertisement from 08:00:c1:00:c1:01
fe80::a00:c1ff:fe00:c101	ff02::1	ICMPv6 Router Advertisement from 08:00:c1:00:c1:01
fe80::a00:c1ff:fe00:c101	ff02::1	ICMPv6 Router Advertisement from 08:00:c1:00:c1:01
fe80::a00:c1ff:fe00:c101	ff02::1	ICMPv6 Router Advertisement from 08:00:c1:00:c1:01
fe80::a00:c1ff:fe00:c101	ff02::1	ICMPv6 Router Advertisement from 08:00:c1:00:c1:01

Figure 12.27

Until a new source appears sending an ICMPv6 Multicast Listener Report Message.

Source	Destination	Message
::	ff02::16	ICMPv6 Multicast Listener Report Message v2

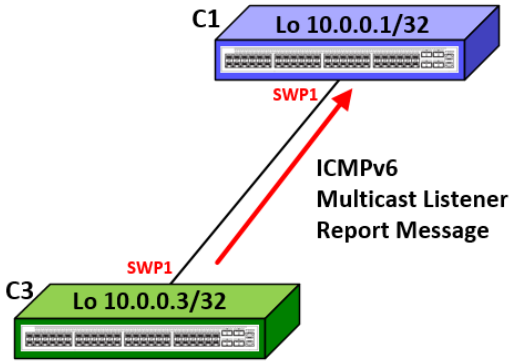


Figure 12.28

This message is being sourced by the Cumulus box C3 (a leaf) on its interface swp1 as observed in the next snippet.

MAC address of C3's interface SWP1

```
Ethernet II, Src: 08:00:c3:00:c3:01 Dst: IPv6mcast_16 (33:33:00:00:00:16)
Internet Protocol Version 6, Src: ::, Dst: ff02::16
Internet Control Message Protocol v6
  Type: Multicast Listener Report Message v2 (143)
  Code: 0
  Checksum: 0xa96e [correct]
  Reserved: 0000
  Number of Multicast Address Records: 2
▷ Multicast Address Record Changed to exclude: ff02::1:ff00:c301
▷ Multicast Address Record Changed to exclude: ff02::2
```

Link Local Address of C3's interface SWP1

Figure 12.29

Followed by an ICMPv6 Neighbor Solicitation Message which contains the Link Local Address of Cumulus box C3 for interface swp1.

```
::      ff02::1:ff00:c301  ICMPv6 Neighbor Solicitation for fe80::a00:c3ff:fe00:c301
```

```
Ethernet II, Src: 08:00:c3:00:c3:01 Dst: IPv6mcast_ff:00:c3:01 (33:33:ff:00:c3:01)
Internet Protocol Version 6, Src: ::, Dst: ff02::1:ff00:c301
Internet Control Message Protocol v6
  Type: Neighbor Solicitation (135)
  Code: 0
  Checksum: 0x2a23 [correct]
  Reserved: 00000000
  Target Address: fe80::a00:c3ff:fe00:c301
```

Figure 12.30

Cumulus C3 also advertises a Link Layer Discovery Protocol LLDP.

```
Ethernet II, Src: 08:00:c3:00:c3:01  Dst: LLDP_Multicast (01:80:c2:00:00:0e)
Link Layer Discovery Protocol
▷ Chassis Subtype = MAC address, Id: 08:00:27:c9:b2:c9
▷ Port Subtype = Interface name, Id: swp1
▷ Time To Live = 120 sec
▷ System Name = C3
▷ System Description = Cumulus Linux
▷ Capabilities
▷ Management Address
▷ Port Description = swp1
▷ End of LLDPDU
```

Figure 12.31

The two cumulus boxes C1 and C3 can "see" each other at this point. They have mutually discovered as they know their corresponding Link Local Addresses.

Source	Destination	Message	
fe80::a00:c1ff:fe00:c101	ff02::1	ICMPv6 Router Advertisement from 08:00:c1:00:c1:01	C1 Spine
fe80::a00:c3ff:fe00:c301	ff02::1	ICMPv6 Router Advertisement from 08:00:c3:00:c3:01	C3 Leaf

Figure 12.32

Hence, they can start a transport layer session now and that begins with a TCP 3-way handshake. The port number 179 belongs to BGP.

	Source	Destination	Protocol	Exchange
C3	fe80::a00:c3ff:fe00:c301	fe80::a00:c1ff:fe00:c101	TCP	59042 → 179 [SYN]
C1	fe80::a00:c1ff:fe00:c101	fe80::a00:c3ff:fe00:c301	TCP	179 → 59042 [SYN, ACK]
C3	fe80::a00:c3ff:fe00:c301	fe80::a00:c1ff:fe00:c101	TCP	59042 → 179 [ACK]

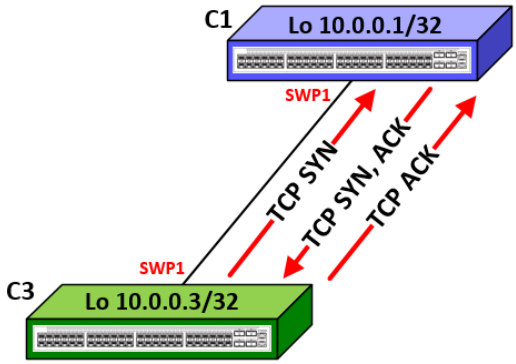


Figure 12.33

Once the transport session has been set up by TCP, the BGP conversation starts with Cumulus 03 (Leaf) sending a BGP OPEN message. Immediately, C1 also sends its corresponding OPEN.

	Source	Destination	Protocol	Message
C3	fe80::a00:c3ff:fe00:c301	fe80::a00:c1ff:fe00:c101	BGP	OPEN Message
C1	fe80::a00:c1ff:fe00:c101	fe80::a00:c3ff:fe00:c301	BGP	OPEN Message

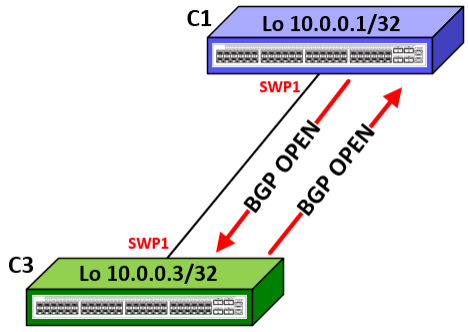
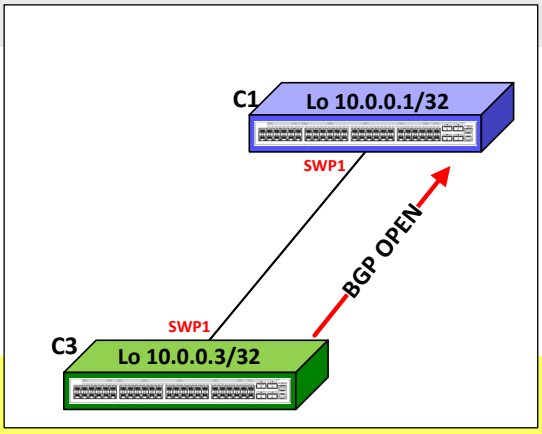


Figure 12.34

It is important to analyze the OPEN message in detail because this message is crucial for the correct functioning of this BGP model. This snippet shows the BGP OPEN message sent from C3 to C1 with its complete encapsulation from Layer 2 up to BGP.

Border Gateway Protocol BGP



Marker FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF

Length 97 Bytes

Message Type: **OPEN** (1)

My AS: **65003**

Hold Time: 3

BGP Identifier: **10.0.0.3**

Optional Parameters Length: 68

Optional Parameters:

- Capability Multiprotocol Extensions AFI IPv4 SAFI Unicast
- Capability Multiprotocol Extensions AFI IPv6 SAFI Unicast
- **Capability Extended Next Hop Encoding**
- **Capability Support for additional paths**
- **Capability FQDN (hostname C3)**
- Capability Route Refresh
- Capability Route Refresh
- Capability Support for Autonomous System Number of 4 Bytes
- Capability Support for graceful restart

TCP source Port 59042, destination port 179 (BGP)

IPv6 source FE80::0A00:C3FF:FE00:C301, destination FE80::0A00:C1FF:FE00:C101

Ethernet Frame source MAC 08:00:C3:00:C3:01, destination MAC 08:00:C1:00:C1:01

Figure 12.35

The following observations are significant in this analysis:

- The BGP message is delivered by an IPv6 packet.

IPv6 source FE80::0A00:C3FF:FE00:C301, destination FE80::0A00:C1FF:FE00:C101

- The IPv6 source and destination addresses are the Auto configured Link Local addresses. For that reason, there is no need to specify in the configuration of the routers the destination address of the BGP peer. This is one of the advantages of this deployment.
- The router identifier of C3 is the loopback IPv4 address 10.0.0.3.
- The Hold Time is only 3 seconds. In this deployment, the Keepalive rate has been set to 1 message per second. Missing three Keepalives from a peer router means that such neighbor is down. The default value for a Cisco router is 180 seconds and for a Juniper router is 90 seconds. In the data centre is important to detect dead neighbours faster.
- The router describes 9 capabilities, most of them are typical. However, there are three capabilities that merit a more detailed look into them. They are the Extended Next Hop Encoding, Support for Additional Paths, and Fully Qualified Domain Name capabilities.

Generally, routes advertised by BGP have a Next Hop that belongs to the same Address Family. However, in this implementation of a Leaf and Spine topology, the next hop address belongs to the IPv6 Address family while the destination prefixes belong to the IPv4 Address Family. Even though that sounds like an incompatibility, RFC 5549 [7] actually describes the use of the Extended Next Hop Encoding for Multiprotocol-BGP (MP-BGP) which allows for an IPv6 address to be the Next Hop for IPv4 destinations. This snippet of the capability dissects this function.

```
◢ Optional Parameter: Capability
    Parameter Type: Capability (2)
    Parameter Length: 8
  ◢ Capability: Extended Next Hop Encoding
      Type:   Extended Next Hop Encoding (5)
      Length: 6
      Value:  0001        0001              0002
                ⇩           ⇩                 ⇩
          AFI:IPv4(1)  SAFI:Unicast (1)   Next Hop AFI:IPv6(2)
```

Figure 12.36

The value of the capability 0001 0001 0002 is a triplet of 6 bytes as specified in RFC-5549 [7]. AFI stands for Address Family Identifier while SAFI is Subsequent AFI which can be either Unicast (1) or Multicast (2). In the current case, the BGP OPEN is notifying that the router understands that IPv4 Unicast addresses could have an IPv6 address as the Next Hop.

Another important capability is Support for Additional Paths. This capability is still in stage of development [8] and it refers that a BGP speaker can advertise multiple paths to the same destination prefix. In default mode, BGP can only advertise the route as it is installed in the IP routing table. The following snippet shows this Optional Capability. Interestingly, the sniffer application (Wireshark) has not been yet updated to understand its value.

```
Optional Parameter: Capability
  Parameter Type: Capability (2)
  Parameter Length: 10
◢ Capability: Support for Additional Paths
    Type: Support for Additional Paths (69)
  ▷ Length: 8
    Unknown: 0001010100020101
```

Figure 12.37

The OPEN message also includes an interesting new capability [11] actually being developed by Cumulus specifically for the data centre. This is to advertise the Fully Qualified Domain Name (FQDN) of the router to simplify troubleshooting because it is easier to remember a meaningful name than an IP address. This FQDN can be registered in a DNS naming domain and they can be used for monitoring and configuration purposes also.

```
◢ Optional Parameter: Capability
    Parameter Type: Capability (2)
    Parameter Length: 6
  ◢ Capability: FQDN Capability
      Type: FQDN Capability (73)
      Length: 4
      Hostname Length: 2
      Hostname: C3
      Domain Name Length: 0
      Domain Name:
```

Figure 12.38

Finally, the following snippets display the advertisement of the Multiprotocol Capabilities for IPv4 Unicast (AFI 1 SAFI 1) and IPv6 Unicast (AFI 2 SAFI 1. This is how the routers advertise the Address Families that they support.

```
◢ Capability: Multiprotocol extensions capability
    Type: Multiprotocol extensions capability (1)
    Length: 4
    AFI: IPv4 (1)
    SAFI: Unicast (1)
```

```
◢ Capability: Multiprotocol extensions capability
    Type: Multiprotocol extensions capability (1)
    Length: 4
    AFI: IPv6 (2)
    SAFI: Unicast (1)
```

Figure 12.39

The exchange of OPEN messages acts as presentation cards for the routers involved. If there are no major errors detected in the information, the routers proceed to the next stage which is to exchange the actual routing information with BGP UPDATE messages.

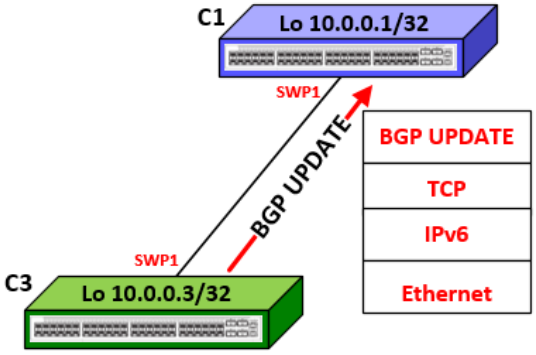


Figure 12.40

```
Ethernet II, Src: 08:00:c3:00:c3:01 (C3)  Dst: 08:00:c1:00:c1:01 (C1)
Internet Protocol Version 6, Src: fe80::a00:c3ff:fe00:c301, Dst: fe80::a00:c1ff:fe00:c101
Transmission Control Protocol, Src Port: 59042 Dst Port: 179
Border Gateway Protocol - UPDATE Message
  Marker: ffffffffffffffffffffffffffffffff
  Length: 94
  Type: UPDATE Message (2)
  Withdrawn Routes Length: 0
  Total Path Attribute Length: 71
◢ Path attributes
  ◢ Path Attribute - MP_REACH_NLRI
    ▷ Flags: 0x90, Optional, Length: Optional, Non-transitive, Complete, Extended Length
      Type Code: MP_REACH_NLRI (14)
      Length: 46
      Address family identifier (AFI): IPv4 (1)
      Subsequent address family identifier (SAFI): Unicast (1)
    ◢ Next hop network address (32 bytes)
        Next Hop: fe 80 00  00 00 00 00 00 0a 00 c3 ff fe 00 c3 01
      Number of Subnetwork points of attachment (SNPA): 0
    ◢ Network layer reachability information (9 bytes)
      ▷ 10.0.0.3/32
      ▷ 172.16.3.0/24 ⇦
  ▷ Path Attribute - ORIGIN: IGP
  ▷ Path Attribute - AS_PATH: 65003
  ▷ Path Attribute - MULTI_EXIT_DISC: 0
```

(This address is repeated thus 32 bytes)

Figure 12.41

The previous snippet shows that Router C3 is sending the following UPDATE advertising prefix 172.16.3.0/24 located in AS65003. The next hop for this prefix is set to the IPv6 Link Local for interface swp1. When C1 receives this information, it will install it in its routing table and then it will propagate the prefix to the other autonomous systems AS65004 and AS65005 hence achieving reachability within the data centre.

Chapter Summary

The Clos Leaf and Spine model developed by Cumulus Linux is a testimony to the flexibility and versatility of the BGP protocol. The developers of this model reveal an insight into the functioning of IP networks and especially into the inner mechanics of BGP. By adapting existing protocols and modifying some aspects of others, a brilliant solution that fits the necessity of data centres have been pieced together.

Chapter References

[1] Charles Clos. (1952). A Study of Non-Blocking Switching Networks. Bell System Technical Journal. https://archive.org/details/bstj32-2-406

[2] K. Lougheed, Y. Rekhter. (1989). RFC 1105 Border Gateway Protocol. http://tools.ietf.org/html/rfc1105

[3] Mohammed Baig. (2016). Tutorial: Configuring a BGP Unnumbered Leaf and Spine Topology with Cumulus VX and GNS3. Sheridan College. www.codewerx.ca

[4] IEEE 802.1AB-(2009) - IEEE Standard for Local and Metropolitan Area Networks-- Station and Media Access Control Connectivity Discovery.

[5] R. Vida, L. Costa. (2004). RFC 3810 Multicast Listener Discovery version 2 (MLDv2) for IPv6. https://tools.ietf.org/html/rfc3810

[6] T. Narten, E. Nordmark, W. Simpson, H. Soliman. (2007). RFC 4861 Neighbor Discovery for IP version 6 (IPv6). https://tools.ietf.org/html/rfc4861

[7] F. Le Faucheur. E. Rosen. RFC 5549. (2009). Advertising IPv4 Network Layer Reachability Information with an IPv6 Next Hop. https://tools.ietf.org/html/rfc5549

[8] D. Walton, A. Retana, E. Chen, J. Scuder. (2014). Draft Advertisement of Multiple Paths in BGP. https://tools.ietf.org/html/draft-ietf-idr-add-paths-10

[9] RFC 5492. J. Scudder. R. Chandra. (2009). Capabilities Advertisement with BGP-4. https://tools.ietf.org/html/rfc5492

[10] Quagga Routing Suite. (2016) http://www.nongnu.org/quagga

[11] D. Walton, D. Dutt. (2016). Internet-Draft Hostname Capability for BGP. https://tools.ietf.org/html/draft-walton-bgp-hostname-capability-02

Appendix

Configuration Node C1

/etc/hostname

```
C1
```

/etc/hosts

```
127.0.1.1    C1  C1
```

/etc/network/interfaces

```
# Management Interface
auto eth0
iface eth0 inet dhcp

# Loopback interface
auto lo
iface lo inet loopback
address 10.0.0.1/32

# Switchport interface SWP1
auto swp1
iface swp1
address 10.0.0.1/32

# Switchport interface SWP2
auto swp2
iface swp2
address 10.0.0.1/32

# Switchport interface SWP3
auto swp3
iface swp3
address 10.0.0.1/32
```

/etc/quagga/Quagga.conf

```
interface swp1
no ipv6 nd suppress-ra
ipv6 nd ra-interval 3

interface swp2
no ipv6 nd suppress-ra
ipv6 nd ra-interval 3

interface swp3
no ipv6 nd suppress-ra
ipv6 nd ra-interval 3

router bgp 65000
bgp router-id 10.0.0.1
network 10.0.0.1
maximum-paths 64
bgp bestpath as-path multipath-relax
neighbor LEAF peer-group
neighbor LEAF description "SPINE peers with LEAF"
neighbor LEAF remote-as external
neighbor LEAF advertisement-interval 0
neighbor LEAF timers 1 3
neighbor LEAF capability extended-nexthop
neighbor swp1 interface
neighbor swp2 interface
neighbor swp3 interface
neighbor swp1 peer-group LEAF
neighbor swp2 peer-group LEAF
neighbor swp3 peer-group LEAF
address-family ipv6 unicast
neighbor LEAF activate
neighbor swp1 activate
neighbor swp2 activate
neighbor swp3 activate
```

Configuration Node C2

/etc/hostname

```
C2
```

/etc/hosts

```
127.0.1.1    C2  C2
```

/etc/network/interfaces

```
# Management Interface
auto eth0
iface eth0 inet dhcp

# Loopback interface
auto lo
iface lo inet loopback
address 10.0.0.2/32

# Switchport interface SWP1
auto swp1
iface swp1
address 10.0.0.2/32

# Switchport interface SWP2
auto swp2
iface swp2
address 10.0.0.2/32

# Switchport interface SWP3
auto swp3
iface swp3
address 10.0.0.2/32
```

/etc/quagga/Quagga.conf

```
interface swp1
no ipv6 nd suppress-ra
ipv6 nd ra-interval 3

interface swp2
no ipv6 nd suppress-ra
ipv6 nd ra-interval 3

interface swp3
no ipv6 nd suppress-ra
ipv6 nd ra-interval 3

router bgp 65000
bgp router-id 10.0.0.2
network 10.0.0.2
maximum-paths 64
bgp bestpath as-path multipath-relax
neighbor LEAF peer-group
neighbor LEAF description  "SPINE peers with LEAF"
neighbor LEAF remote-as external
neighbor LEAF advertisement-interval 0
neighbor LEAF timers 1 3
neighbor LEAF capability extended-nexthop
neighbor swp1 interface
neighbor swp2 interface
neighbor swp3 interface
neighbor swp1 peer-group LEAF
neighbor swp2 peer-group LEAF
neighbor swp3 peer-group LEAF

address-family ipv6 unicast
neighbor LEAF activate
neighbor swp1 activate
neighbor swp2 activate
neighbor swp3 activate
```

Configuration Node C3

/etc/hostname C3 **/etc/hosts** 127.0.1.1 C3 C3 **/etc/network/interfaces** # Management Interface auto eth0 iface eth0 inet dhcp # Loopback interface auto lo iface lo inet loopback address 10.0.0.3/32 # Switchport interface SWP1 auto swp1 iface swp1 address 10.0.0.3/32 # Switchport interface SWP2 auto swp2 iface swp2 address 10.0.0.3/32	**/etc/quagga/Quagga.conf** interface swp1 no ipv6 nd suppress-ra ipv6 nd ra-interval 3 interface swp2 no ipv6 nd suppress-ra ipv6 nd ra-interval 3 router bgp 65003 bgp router-id 10.0.0.3 network 10.0.0.3 network 172.16.3.0/24 maximum-paths 64 bgp bestpath as-path multipath-relax neighbor SPINE peer-group neighbor SPINE description "LEAF peers with SPINE" neighbor SPINE remote-as external neighbor SPINE advertisement-interval 0 neighbor SPINE timers 1 3 neighbor SPINE capability extended-nexthop neighbor swp1 interface neighbor swp2 interface neighbor swp1 peer-group LEAF neighbor swp2 peer-group LEAF address-family ipv6 unicast neighbor SPINE activate neighbor swp1 activate neighbor swp2 activate

Configuration Node C4

/etc/hostname

```
C4
```

/etc/hosts

```
127.0.1.1    C4  C4
```

/etc/network/interfaces

```
# Management Interface
auto eth0
iface eth0 inet dhcp

# Loopback interface
auto lo
iface lo inet loopback
address 10.0.0.4/32

# Switchport interface SWP1
auto swp1
iface swp1
address 10.0.0.4/32

# Switchport interface SWP2
auto swp2
iface swp2
address 10.0.0.4/32
```

/etc/quagga/Quagga.conf

```
interface swp1
no ipv6 nd suppress-ra
ipv6 nd ra-interval 3

interface swp2
no ipv6 nd suppress-ra
ipv6 nd ra-interval 3

router bgp 65004
bgp router-id 10.0.0.4
network 10.0.0.4
network 172.16.4.0/24
maximum-paths 64
bgp bestpath as-path multipath-relax
neighbor SPINE peer-group
neighbor SPINE description  "LEAF peers with SPINE"
neighbor SPINE remote-as external
neighbor SPINE advertisement-interval 0
neighbor SPINE timers 1 3
neighbor SPINE capability extended-nexthop
neighbor swp1 interface
neighbor swp2 interface
neighbor swp1 peer-group LEAF
neighbor swp2 peer-group LEAF

address-family ipv6 unicast
neighbor SPINE activate
neighbor swp1 activate
neighbor swp2 activate
```

Configuration Node C5

/etc/hostname C5 **/etc/hosts** 127.0.1.1 C5 C5 **/etc/network/interfaces** # Management Interface auto eth0 iface eth0 inet dhcp # Loopback interface auto lo iface lo inet loopback address 10.0.0.5/32 # Switchport interface SWP1 auto swp1 iface swp1 address 10.0.0.5/32 # Switchport interface SWP2 auto swp2 iface swp2 address 10.0.0.5/32	**/etc/quagga/Quagga.conf** interface swp1 no ipv6 nd suppress-ra ipv6 nd ra-interval 3 interface swp2 no ipv6 nd suppress-ra ipv6 nd ra-interval 3 router bgp 65005 bgp router-id 10.0.0.5 network 10.0.0.5 network 172.16.5.0/24 maximum-paths 64 bgp bestpath as-path multipath-relax neighbor SPINE peer-group neighbor SPINE description "LEAF peers with SPINE" neighbor SPINE remote-as external neighbor SPINE advertisement-interval 0 neighbor SPINE timers 1 3 neighbor SPINE capability extended-nexthop neighbor swp1 interface neighbor swp2 interface neighbor swp1 peer-group LEAF neighbor swp2 peer-group LEAF address-family ipv6 unicast neighbor SPINE activate neighbor swp1 activate neighbor swp2 activate